



EDUCACIÓN
SECRETARÍA DE EDUCACIÓN PÚBLICA



ITESCA[®]
*Instituto Tecnológico
Superior de Cajeme*

INSTITUTO TECNOLÓGICO SUPERIOR DE CAJEME

**Desarrollo de un controlador para un
Robot CRS de 6 Grados De Libertad**

TESIS

**QUE PARA OBTENER EL GRADO DE
MAESTRO EN INGENIERIA MECATRONICA**

PRESENTA

ING. ALAN ALFREDO CHACÓN VIGUERÍA

Director de Tesis:

DR. JOSÉ EFRÉN RUELAS RUIZ

Fecha

25-05-2022

No. de Registro

MIM-67

Dr. José Efrén Ruelas Ruiz
Maestría en Ingeniería Mecatrónica,
PRESENTE

A través de este documento, me permito comunicarle que el alumno

Alan Alfredo Chacón Viguería

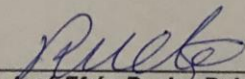
cuyo proyecto de obtención de grado

Desarrollo de un controlador para un Robot CRS de 6 Grados De Libertad

Cumple satisfactoriamente, con los Lineamientos para la Operación de Estudios de Posgrado en el Sistema Nacional de Educación Superior Tecnológica, por lo que ha sido aceptado para la obtención del grado:

Maestro en Ingeniería Mecatrónica

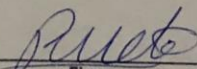
Sin más por el momento, quedo de Usted.


Dr. José Efrén Ruelas Ruiz
Director de proyecto de obtención de grado

25-05-2022
Fecha

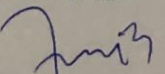
DICTAMEN DE ACEPTACION - COMISIÓN REVISORA

PRESIDENTE Dr. José Efrén Ruelas Ruiz
Nombre


Firma

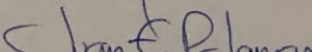
25-05-2022
Fecha

MIEMBRO 1 Dr. Francisco Javier Ochoa Estrella
Nombre


Firma

25-05-2022
Fecha

MIEMBRO 2 Dr. Juan Enrique Palomares Ruiz
Nombre


Firma

25-05-2022
Fecha

“Agradezco a mi familia, a mi asesor y en general a los profesores y personal del ITESCA al igual que a mis compañeros y especialmente a CONACYT por su apoyo durante el posgrado”

INDICE

CAPITULO I	6
INTRODUCCIÓN	6
1.1. Antecedentes	7
1.2. Planteamiento del problema	9
1.3. Justificación	9
1.4. Objetivos	10
1.4.1. Objetivo general	10
1.4.2. Objetivos específicos	10
1.5. Alcances y limitaciones	11
1.5.1. Alcances	11
1.5.2. Limitaciones	11
CAPITULO II	12
MARCO TEORICO	12
2.1. Robótica	12
2.1.1. Definición de Robótica y robot	12
2.1.2. Encoder	13
2.1.3. Servo motor	15
2.1.4. Fuente de poder	16
2.2. Controladores de robots	17
2.2.1. Definición de Controlador	17
2.2.2. Tipos de Controladores	18
2.2.3. Microcontroladores	20
2.3. Cinemática y rotaciones	22
2.3.1. Grado de libertad	23
2.3.2. Modelo cinemático	23
2.3.3. Cuaterniones	25
2.3.4. Arquitecturas	27
2.3.7. Modelado cinemático CRS A465	29
CAPITULO III	31
METODOLOGÍA.....	31
3.1 Descripción del sistema a desarrollar	31
3.2 Materiales y equipo	32
3.2.1. Caracterización del robot	33
3.2.2. Cálculo de la cinemática directa	35
3.2.3. Caracterización de la fuente de poder	40
3.2.4. Caracterización de los conectores	42
3.2.5. Caracterización del puente H	43

3.2.6. Caracterización del microcontrolador	44
3.3 Desarrollo de Interface Arduino-Robot.....	45
3.3.1 Programación de la interfaz	46
3.3.2 Conexiones	47
3.4 Interface Arduino-Java.....	49
3.5 Interface Java-Usuario	50
CAPITULO IV	52
RESULTADOS	52
4.1 Cinemática del robot.....	52
4.2 Interface Arduino-Robot	55
4.2.1 Programación de la interfaz	55
4.2.2 Conexiones	59
4.3 Interface Arduino-Java.....	61
4.4 Resultados experimentales	68
CAPITULO V	71
CONCLUSIONES.....	71
5.1 Conclusiones del proyecto	71
5.2 Futuros proyectos	72
ANEXOS	73
Anexo A: Tabla de conexiones del CPC-24 para los motores	73
Anexo B: Tabla de conexiones del CPC-57 para los encoders	73
Anexo C: Códigos empleados en la programación del controlador	74
C.1: Interrupciones del Arduino.....	74
C.2: Actualizar la posición.....	75
C.3: Señales del controlador al puente H.....	75
C.4: Función para mandar a posición deseada	76
C.5: Confirmación de la comunicación Arduino	78
C.6: Confirmación de la comunicación Java	78
C.7: Envío de caracteres a Arduino desde Java	79
C.8: Envío de una nueva posición desde Java	79
C.9: Lectura de una nueva posición desde Arduino	80
Anexo D: Código de letras para la comunicación Java-Arduino	82
Anexo E: Representación esquemática completa del circuito eléctrico	84
REFERENCIAS BIBLIOGRAFICAS	85

INDICE DE FIGURAS

Figura 1.1 Robot CRS A465 junto al controlador C500C y su terminal	7
Figura 2.1 Ejemplos de Robots	13
Figura 2.2 Encoder Incremental.....	14
Figura 2.3 Encoder incremental a) principio básico b) pistas concéntricas.....	14
Figura 2.4 Encoder absoluto	15
Figura 2.5 Ejemplo de servomotor y estructura interna	15
Figura 2.6 Ejemplo de fuentes de poder para actuadores hidráulicos, eléctricos y neumáticos	16
Figura 2.7 Ilustración de un controlador implementado.....	18
Figura 2.8 Ilustración de un robot y su controlador dual	19
Figura 2.9 Robot conectado a un microcontrolador.....	19
Figura 2.10 Esquematación general de un microcontrolador	20
Figura 2.11 Familias de microcontroladores	21
Figura 2.12 Tarjetas de microcontroladores.....	22
Figura 2.13 Ejemplificación de los grados de libertad en un robot	23
Figura 2.14 a) Movimiento traslacional b) Movimiento rotacional	24
Figura 2.15 Ejemplo de una esquematización para el análisis cinemático.....	24
Figura 2.16 Tipos de articulaciones a) Rotacional, b) Prismática.....	27
Figura 2.17 Localización de las juntas de rotación en el CRS A465	29
Figura 2.18 Localización de los últimos tres grados de libertad del CRS A465, conocidos como yaw, pitch & roll.....	30
Figura 2.19 Declaración de las bases utilizando la regla de la mano derecha.....	30
Figura 3.1 Metodología para el desarrollo de un controlador	31
Figura 3.2 Solución propuesta	32
Figura 3.3 Brazo robot articulado A465.....	33
Figura 3.4 Rango de movimiento y dimensiones del A465 a) Ancho del robot b) Área de trabajo superior c) Área de trabajo lateral	34
Figura 3.5 Rango de movimiento del efector final	35
Figura 3.6 Consideraciones de los giros.....	37
Figura 3.7 Consideraciones de la posición Inicial.....	38
Figura 3.8 Fuente de poder del C500C	40
Figura 3.9 Puente H DC Mosfet IRF3205	43
Figura 3.10 Arduino Due.....	44
Figura 3.11 Motor-Encoder incremental con Arduino.....	45
Figura 3.12 Esquematación de conexiones para dos motores.....	47
Figura 3.13 Esquematación de conexiones para la electroválvula.....	48
Figura 3.14 Arduino con Java.....	49
Figura 3.15 Interfaz con Java	50
Figura 3.16 Interface COSIMIR.....	51
Figura 4.1 Configuración Home (175°,90°,200°,180°,105°,180°).....	52
Figura 4.2 Posición Vertical (175°,90°,110°,180°,105°,180°).....	53
Figura 4.3 Configuración de muestra (200°,110°,200°,150°,130°,220°).....	53
Figura 4.4 Pulsos e Interrupciones.....	56
Figura 4.5 Diagrama de bloques	57
Figura 4.6 Función Home	59
Figura 4.7 Diagrama eléctrico	60
Figura 4.8 Conexiones del controlador.....	61
Figura 4.9 Comunicación Arduino Java.....	62
Figura 4.10 Ventana principal.....	64
Figura 4.11 Ventana de Ayuda.....	65
Figura 4.12 Ventana de conexiones.....	65
Figura 4.13 Área de Trabajo	67

INDICE DE TABLAS

Tabla 2.1 Tipos de Fuentes de poder.....	17
Tabla 2.2 Arquitecturas comunes de robots industriales.....	28
Tabla 3.1 Características del robot A465.....	34
Tabla 3.2 Consideraciones para los cálculos.....	36
Tabla 3.3 Fuente de poder del C500C.....	41
Tabla 3.4 Fuentes A, B y C.....	41
Tabla 3.5 Conexiones y voltajes de los encoders.....	42
Tabla 3.6 Conexiones y voltajes de los motores.....	43
Tabla 3.7 Especificaciones del Arduino Due.....	45
Tabla 3.8 Conexiones de los encoders.....	48
Tabla 4.1 Comparativa de los modelos cinemáticos.....	54
Tabla 4.2 Distribución de los pines.....	60
Tabla 4.3 Comandos programados.....	67
Tabla 4.4 Comparación de características generales.....	68
Tabla 4.5 Ángulos en la posición Home.....	69
Tabla 4.6 Rangos de movimiento.....	69
Tabla 4.7 Error en articulaciones.....	69

CAPITULO I

INTRODUCCIÓN

Los robots y la robótica, en general, adquieren cada vez más una relevancia a nivel global, esta importancia viene debido a la variedad de campos donde tiene aplicación la robótica, abarcando desde la industria alimenticia, hasta la industria médica, pasando por la industria manufacturera y como no, en la educación.

Desde la aparición de Unimate, el primer robot industrial, usado por General Motors en los años 60's, los robots han encontrado variedad de aplicaciones. Los robots realizan trabajos que un ser humano no puede hacer sin exponer su salud o por no tener la habilidad y eficiencia que un robot si puede dar. Estas máquinas llegan a encontrarse en procesos de ensambles de automóviles, cirugías médicas y centros de investigación por dar ejemplos. El diseño de un robot dependerá de su área de trabajo, a partir de ella se harán los requerimientos del hardware y de software.

En el presente trabajo de tesis se expone el diseño y desarrollo de un controlador para un Robot de 6 GDL fabricado por CRS, la tesis pertenece a la especialidad del diseño mecatrónico.

El desarrollo del controlador toma en cuenta el diseño mecánico del robot que establece las características de su movimiento. Dicho controlador también considera la teoría electrónica para el manejo de la potencia a suministrar a los elementos electromecánicos del robot como lo son los motores y los encoders, todo en sinergia con la programación que se implementará en un microcontrolador y un computador. El computador llevará a cabo los cálculos y almacenará los datos necesarios para mover el robot además brindará una interfaz al usuario para ingresar las instrucciones y monitorear el estado de las operaciones.

1.1. Antecedentes

Fundada en 1982, CRS Robotics Corporation fue una compañía de diseño, manufactura, distribución y servicio de robots articulados, uno de sus productos fue el Brazo Robot CRS A465 que consistía en un modelo articulado con seis grados de libertad, del cual dispone el Instituto Tecnológico Superior de Cajeme (ITESCA) pero se encuentra en desuso debido a la obsolescencia del controlador.

El Brazo Robot CRS A465 era comercializado junto con el Sistema de control C500C, véase la Figura 1.1, éste dispositivo permitía el control del brazo mediante una terminal programable, la terminal daba la posibilidad de mover el brazo en tiempo real o escribir un programa con una secuencia de movimientos preestablecida, el C500C era programado usando el lenguaje de programación RAPL-3, este lenguaje permitía el control de varios modelos que eran fabricados por CRS Robotics, entre ellos el A465 [1].



Figura 1.1 Robot CRS A465 junto al controlador C500C y su terminal

Revisando las propuestas de sistemas de control para brazos robot con seis grados de libertad o proyectos semejantes se tiene lo siguiente.

El trabajo de Muhammad Bilal y sus colegas de la University Of Engineering And Technology LAHORE realizado en Mayo del 2018, y titulado como *Design and Control of 6*

DOF Robotic Manipulator (Diseño y control de un Robot Manipulador de 6 GDL), en el se ilustra la esquematización del brazo robot empleando un software de diseño por computadora (SolidWorks), y el desarrollo de un software en el microcontrolador de la tarjeta Tiva C Launchpad Controller de la compañía Texas Instruments, trabajando en el lenguaje de programación C++ con el cual también se agregó una interfaz humano-máquina en una aplicación de escritorio para el robot [2].

Respecto a la reingeniería en sistemas de CRS Robotics está la tesis presentada en mayo del 2016 de Francisco Javier García Gutiérrez titulada *Desarrollo y construcción de una tarjeta para la adquisición de datos y control de robots cooperativos* en la Universidad Autónoma de México; en ella se analizó el sistema de control C500C, se capturó y analizó las señales de los distintos cables que conforman el controlador y como estas manipulaban cada una de las articulaciones de los modelos A255 y A465 [3].

Un trabajo enfocado al modelo A465 es el artículo de *Reingeniería del Controlador de un Brazo Robótico de 6 g.d.l.*, publicado por la Revista de Ingeniería Biomédica y Biotecnología en junio de 2018, en él se expone el modelo cinemático del A465 junto con los cálculos de sus trayectorias en cada plano del espacio, al igual que presenta una simulación y una alternativa el control C500C [4].

Por su parte, Carlos Fernando Salinas García, expone en *Modelo Cinemático Directo Del Robot CRS Robotics A465*, del Instituto Tecnológico de Estudios Superiores de la Región Carbonífera un análisis cinemático del robot y se ilustra empleando el software Matlab para su simulación [5].

El trabajo más acercado al presente fue el de Jesus A. Tiburcio Mejia quien en su trabajo de "Reparación de brazo robótico CRS A465 e interfaz gráfica en LabView PIC18F4550" realizó la sustitución de algunas placas del C500C con un microprocesador y creo una interfaz gráfica para controlarla desde un computador [6].

Finalmente, el L.M. Juan Enrique Palomares Ruiz y su trabajo de *Modelación Cinemática Del Robot CRS A465, Utilizando El Álgebra De Quaterniones*; establece las ecuaciones de movimiento y velocidad del A465 a través del cálculo de Quaterniones resolviendo el problema de la cinemática directa e inversa del robot, lo que representa un avance dado que con dicha metodología será posible obtener su posición en la simulación [7].

1.2. Planteamiento del problema

Actualmente el Instituto Tecnológico Superior de Cajeme (ITESCA) posee dos robots CRS de seis grados de libertad en desuso debido a la falta del controlador el cual se encuentra desactualizado. Esta problemática se encuentra presente en varias instituciones del país además del ITESCA.

1.3. Justificación

La realización del presente proyecto permitirá ampliar el campo de restauración y renovación de tecnologías abandonadas, las cuales suelen comúnmente terminar en instituciones educativas; al poder reactivar un equipo de arquitectura cerrada se obtendrá una metodología que se pueda reutilizar para casos similares, además de que mediante la incursión del proyecto se llegaran a exponer una diversidad de factores críticos para el correcto funcionamiento de los robots en general.

La posibilidad de volver a emplear equipo en desuso mediante alguna reestructuración puede otorgar un ahorro de recursos financieros cuando el procedimiento es llevado de manera interna, debido a que al querer llevar esto a cabo en esta clase de máquinas se suelen considerar únicamente dos opciones, la contratación de un tercero para reparar el sistema, incluyendo posiblemente costos de formación y un mantenimiento cobrado adicionalmente, o la directa compra de un equipo nuevo, dos opciones caras en comparación.

Debido a que toda la metodología será redacta y se pretenden crear los sistemas de software de manera que sean de acceso público también se disminuiría el gasto en

actualizaciones, adicional a esta ventaja, gracias a la facilidad para obtener estos archivos permitirá a las instituciones educativas que posean equipos similares involucrar a los alumnos en una mejora continua apoyando su desarrollo mediante modificaciones, creación de aplicaciones o con el simple análisis de la misma documentación.

De igual manera si se considerará optar por dejar el equipo en desuso se estaría desperdiciando la posibilidad de mejorar el nivel académico de los alumnos que pudiesen hacer uso de él, dejándolos por debajo de instituciones que cuentan con estos dispositivos y por ende haciendo a las propias escuelas menos atractivas para el ingreso a carreras relacionadas a esta área.

1.4. Objetivos

Los objetivos que se persiguen en este trabajo de tesis son los siguientes:

1.4.1. Objetivo general

Desarrollar un controlador que permita el manejo del robot CRS modelo A465 de seis grados de libertad.

1.4.2. Objetivos específicos

- 1) Caracterizar eléctrica y mecánicamente el brazo A465.
- 2) Seleccionar e instrumentar los componentes del controlador como microcontrolador, servomotores, encoders y circuitería electrónica.
- 3) Desarrollar el software de comunicación y control del robot A465.
- 4) Desarrollar una interface de usuario.
- 5) Incorporar los modelos cinemáticos al robot en tiempo real.
- 6) Configurar el microcontrolador y el computador para el control.
- 7) Agregar medidas de seguridad del robot.

1.5. Alcances y limitaciones

Los alcances y límites de este proyecto son los siguientes:

1.5.1. Alcances

- 1) Diseñar y construir un prototipo del controlador e interface que sea programable por el usuario.
- 2) Utilizar para el control un equipo comercial como el Arduino Due.
- 3) Algoritmo enfocado en la funcionalidad básica similar a la de los modelos comerciales.

1.5.2. Limitaciones

- 1) El controlador se limita solo movimientos de rotación y traslación que posea el modelo del brazo disponible, siendo este el A465.
- 2) Se dispone de 2 años para desarrollar y validar el proyecto.
- 3) Emplear circuitos y controladores electrónicos de bajo costo y disponibles a nivel comercial.

CAPITULO II

MARCO TEORICO

En este capítulo se expondrán los conceptos necesarios para el diseño y desarrollo del controlador del robot.

2.1. Robótica

Ambos, el término de 'robótica' y 'robot' se han empleado en variedad de entornos, su uso en los ámbitos novelísticos y cinematográficos dentro de obras del género de la ciencia ficción, o su uso dentro de ambientes industriales y de mercadotecnia en la promoción de novedades o avances puede crear confusión, por lo cual se expondrá la definición usada para dichos términos y otros relacionados en este proyecto.

2.1.1. Definición de Robótica y robot

El término robótica es usado en la tecnología que involucra al diseño, construcción, operación y aplicación de robots. Un robot es un dispositivo que puede ser considerado como una máquina inteligente. Una máquina no inteligente es un dispositivo manual que para actuar requiere de la intervención de un operador durante cada paso del proceso. Una máquina inteligente no requeriría de un ser humano para actuar, pero sí de un computador que lo opere de manera automática, evitando así requerir de una persona que decida que hacer a continuación en cada parte del proceso de acuerdo a las señales del ambiente [8].

Los robots cambiarán su diseño según las necesidades del trabajo a realizar, pudiendo llegar a observarse la similitud con los brazos humanos, aunque otros diseños se alejan de este aspecto, en la Figura 2.1 se ilustran algunos ejemplos de robots, entre ellos existen elementos básicos que se comparten y son los siguientes:

1. Manipulador: El cuerpo principal del robot, la estructura para la manipulación, que consiste de la unión y conexión de los otros elementos estructurales.
2. Órgano efector: Esta conectada al final de las uniones del manipulador y es concerniente a la sujeción de objetos o hacer conexiones a máquinas. Un ejemplo

común es la pinza, consiste de dos dedos los cuales se abren y cierran para tomar objetos y reubicarlos. El término *órgano efector robótico* incluye también dispositivos como cambios de herramientas, pistolas de pintura, pulidores, soldadores de arco, etc.

3. Actuadores: Dispositivos que permiten el movimiento del robot colocados comúnmente en las uniones. Algunos actuadores para robots son los motores eléctricos que rotan una rueda o engranaje y los actuadores lineales, accionados por aire o aceite comprimido, para provocar el movimiento de pistones en los cilindros.
4. Sensores: Usados por el robot para recolectar información acerca del entorno y el propio estado actual del robot. Por ejemplo, los potenciómetros rotatorios son usados para monitorear el ángulo entre dos de las uniones de la mano; sensores de fuerza/tacto son usados en los dedos y la palma de la mano para retroalimentar cuando se ha entrado en contacto con algo, y regular la presión al momento de sujetar un objeto.
5. Controlador: Es la estructura electromecánica que un robot necesita para ser controlado y llevar a cabo las tareas asignadas. Este involucra la información de los sensores, y el proceso de dicha información para enviar comandos a los actuadores y así ejecutar una acción. El controlador se retomará en la sección 2.2 *Controladores de robots*.



Figura 2.1 Ejemplos de Robots

2.1.2. Encoder

Un encoder es un dispositivo que provee una salida digital como resultado de un desplazamiento lineal o angular.

Los encoders pueden ser agrupados en dos categorías:

- Encoders incrementales, los cuales detectan los cambios en la rotación desde una posición anterior.
- Encoders absolutos, los cuales entregan una posición angular en todo momento [8].

1) Encoder incremental

En la Figura 2.2 se muestra un encoder incremental genérico que mide el desplazamiento angular.



Figura 2.2 Encoder Incremental

En el encoder incremental, un haz de luz pasa a través de las rendijas de un disco y es detectado por un sensor de luz. Cuando el disco rota, una salida de pulsos es producida por el sensor con el número de pulsos siendo proporcional al ángulo en el que el disco rota, este mecanismo es ilustrado en la Figura 2.3. La posición del disco, y por lo tanto el ángulo de rotación puede ser determinado por el número de pulsaciones producidas desde que se cambió la posición.

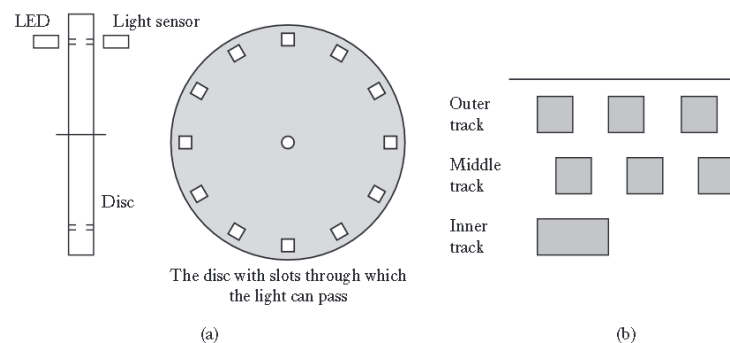


Figura 2.3 Encoder incremental a) principio básico b) pistas concéntricas

2) Encoder absoluto

En la Figura 2.4 se muestra la forma básica de un encoder absoluto para la medición de un desplazamiento angular. Esta entrega una salida en forma de un número binario de varios dígitos, este número representa una posición angular predeterminada. En este ejemplo el disco de rotación tiene tres círculos concéntricos con aberturas y tres sensores para detectar los pulsos de luz. Las aberturas están colocadas de manera secuencial a la salida desde los sensores. El número de bits en binario será igual al número de pistas.

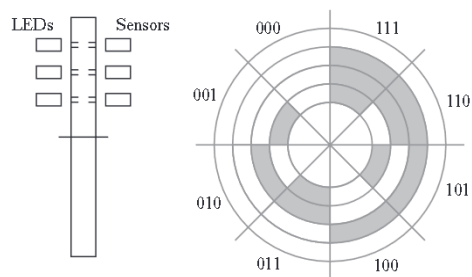


Figura 2.4 Encoder absoluto

2.1.3. Servo motor

Los servomotores son motores impulsados por una corriente que procede de amplificadores electrónicos de corriente directa (CD) o corriente alterna (CA) que poseen su eje de rotación conectado a un conjunto de engranajes para amplificar su torque. Los servomotores de CD varían su tamaño dependiendo de la potencia, aunque en ocasiones son ilustrados como el que se aprecia en la Figura 2.5 [9].



Figura 2.5 Ejemplo de servomotor y estructura interna

Las características fundamentales que se deben buscar en cualquier servomotor de CD o CA son las siguientes:

- 1) Que el par de salida del motor sea aproximadamente proporcional a su voltaje de control aplicado (desarrollado por el amplificador en respuesta a una señal de error).
- 2) Que la dirección del par este determinada por la polaridad instantánea del voltaje de control.

2.1.4. Fuente de poder

La fuente de poder es la encargada de suministrar la energía para que el robot realice su tarea. Existen tres fuentes de poder que pueden llegar a necesitar los robots, de energía eléctrica, hidráulica o neumática como las que se observan en la Figura 2.6, dependiendo del tipo de actuadores del robot.



Figura 2.6 Ejemplo de fuentes de poder para actuadores hidráulicos, eléctricos y neumáticos

Cada fuente de poder posee características únicas y se emplea para distintos ámbitos, la Tabla 2.1 resume los aspectos más importantes [10].

Tabla 2.1 Tipos de Fuentes de poder

Fuente de Poder	Ventajas	Desventajas	Uso
Hidráulica	-Fuerza -Fácil detección de fugas	-Velocidad de respuesta lenta -Fugas de aceite -Ruido	-Manejo de cargas pesadas -Necesidad de grandes fuerzas
Neumática	-Alta velocidad de respuesta -Bajo costo -Limpios	-Ruido -Poca fuerza -Difícil detección de fugas	-Celdas de manufactura -Órganos efectores en robots
Eléctrica	-Limpios -Bajo costo -Fácil de controlar	-Necesitan aditamentos en la mayoría de aplicaciones -Un daño puede ocasionar la pérdida de la fuente	-Prototipos de proyectos -Equipos de precisión -Apoyo como fuente de respaldo

2.2. Controladores de robots

Un controlador decide como actuar ante ciertos eventos y también es el encargado de dar la instrucción a los actuadores, a continuación, se presentarán algunos puntos a tener en cuenta cuando se habla de los estos controladores.

2.2.1. Definición de Controlador

Un sistema de control para un robot humanoide consiste de dos partes: el controlador de los componentes del robot y las instrucciones de movimiento, este último también conocido como controlador de usuario.

El controlador se encarga de recibir los comandos y realizar el proceso para mandarlos al robot usando un algoritmo preestablecido, se genera la trayectoria necesaria y manda los comandos al robot para que este efectúe la orden de movimiento, todo esto conectado como se ve en la Figura 2.7. La capacidad del controlador será definida conforme su

entendimiento del entorno y las capacidades propias del robot, como lo es la evasión de obstáculos o el agarre de objetos respectivamente [11].

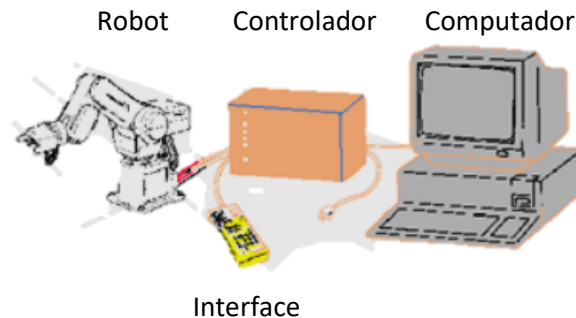


Figura 2.7 Ilustración de un controlador implementado

2.2.2. Tipos de Controladores

En los primeros 'robots', que hoy en día son denominados como autómatas mecánicos, su sistema de control consistía de un conjunto de engranajes o tubos cilíndricos con caminos definidos activados por cuerda, estos sistemas de control estaban diseñados de tal manera que a medida de ir rotando activaban el movimiento de las palancas u otros engranajes que permitían al autómatas cumplir con su proceso.

Hoy en día los robots para procesos industriales y científicos emplean dos tipos de control electrónico, los Controladores Lógicos Programables (PLC's) y los microcontroladores.

Un controlador lógico programable es usado para robots con aplicaciones industriales donde los movimientos están basados en la activación y desactivación de motores y bombas de aire o aceite, no son implementados solos, ya que suelen ser acompañados en conjunto con un controlador adicional de tal manera que el PLC regula el proceso entero y el controlador adicional suple en el control del robot dando como resultado lo que se conoce como un controlador dual el cual se aprecia en la Figura 2.8.

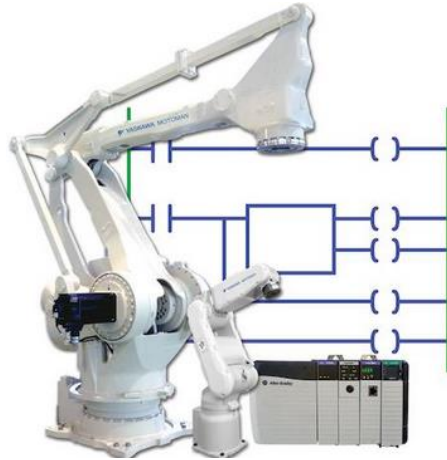


Figura 2.8 Ilustración de un robot y su controlador dual

Estos controladores no son programados con algoritmos que realicen cálculos superiores a los aritméticos, esto debido a la complejidad que tiene su lenguaje de programación para realizar esta tarea, sin embargo, son usados cuando los actuadores requieren de mayor cantidad de potencia de la que un microcontrolador puede dar o cuando el entorno en donde se desarrollan los deja expuestos a golpes, viento, temperaturas extremas, etcétera [8].

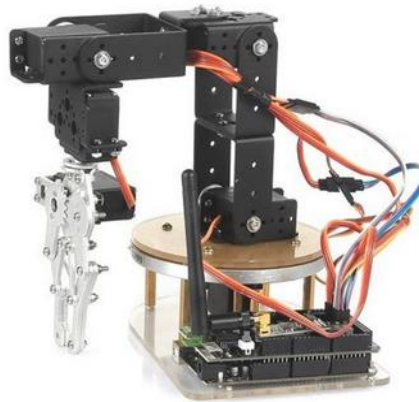


Figura 2.9 Robot conectado a un microcontrolador

Los microcontroladores se tratan de dispositivos que son usados en los sistemas embebidos, se pueden encontrar en aparatos electrónicos como impresoras, lavadoras, juguetes y robots como el que se muestra en la Figura 2.9.

Se emplean microcontroladores para trabajar con prototipados y en aplicaciones orientadas a la investigación y educación más que a su uso industrial, aunque se llegan a encontrar en áreas concretas como electrodomésticos, aeronaves y equipo de trabajo. Su costo es menor al de un PLC y sus lenguajes de programación suelen ser de alto nivel por lo que son para robots que tienen tareas que requieren más procesamiento, además de que están diseñados para añadirles aditamentos [12].

2.2.3. Microcontroladores

Un microcontrolador es la integración de un microprocesador con una memoria e interfaces de entrada y salida, además de otros complementos como temporizadores, todo en un único chip. La Figura 2.11 muestra una esquematización general de un microcontrolador, esta puede variar entre los distintos modelos y marcas [8].

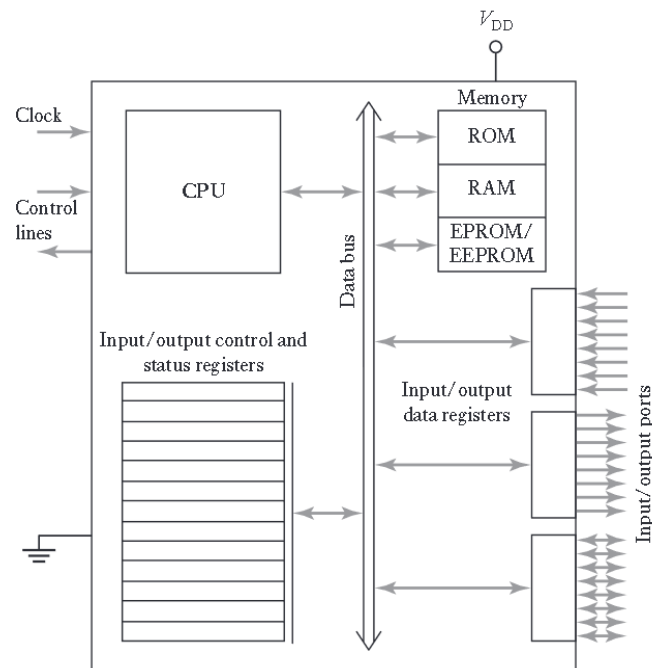


Figura 2.10 Esquematización general de un microcontrolador

1) Familias de microcontroladores

Existen en el mercado una variedad de microcontroladores que cambian su microprocesador, y los aditamentos (la memoria, las entradas/salidas y de más) se seleccionan según la capacidad de procesamiento.

Algunas marcas de microcontroladores actuales en el mercado son las siguientes:

- Motorola cuenta con la familia de microcontroladores 68HCXX.
- Intel tiene el microcontrolador Intel 8051 o los MCS 51.
- Microchip fábrica a los PIC que son microcontroladores que manejan una distinta arquitectura denominada “arquitectura de Harvard” y su familia común son los 16F8X.
- Atmel AVR posee los ATmega328 que se incorporan a las placas Arduino.

Todos estos microcontroladores se ilustran en la Figura 2.12 [12].

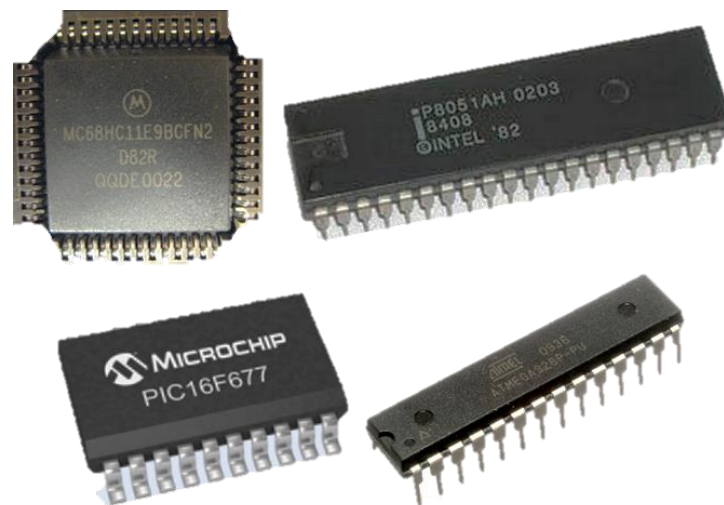


Figura 2.11 Familias de microcontroladores

2) Tarjetas de microcontroladores

Las tarjetas de microcontroladores o placas de desarrollo de hardware consisten en tarjetas de circuitos que cuentan con aditamentos necesarios para trabajar con los microcontroladores ya integrados, incluyendo reguladores, puertos de comunicación, de carga, relojes y otros más dependiendo del modelo.

Entre las tarjetas de microcontroladores comerciales se encuentran las siguientes:

- Placas de Arduino que se programan mediante la IDE propia de Arduino, y posee una gran variedad de distintas placas por ello su uso en prototipos [13].
- Tarjetas ESP desarrolladas por Espressif System integran por defecto módulos que en Arduino son externos como lo son el de wifi o bluetooth, además de que son programables en Microphyton y la IDE de Arduino [14].
- Texas Instruments tienen las placas Hercules que son usadas en proyectos que necesitan precisión y velocidad con los datos a manejar, estas usan el software HALCoGen para su programación [15].
- La Raspberry es un ordenador de placa simple, que se llega a emplear para los mismos propósitos que las otras placas de desarrollo, está dirigida para proyectos que requieran potencia de procesamiento como en el caso de necesitar usar un sistema operativo.

Podemos ver ejemplos de cada placa de desarrollo en la Figura 2.13.

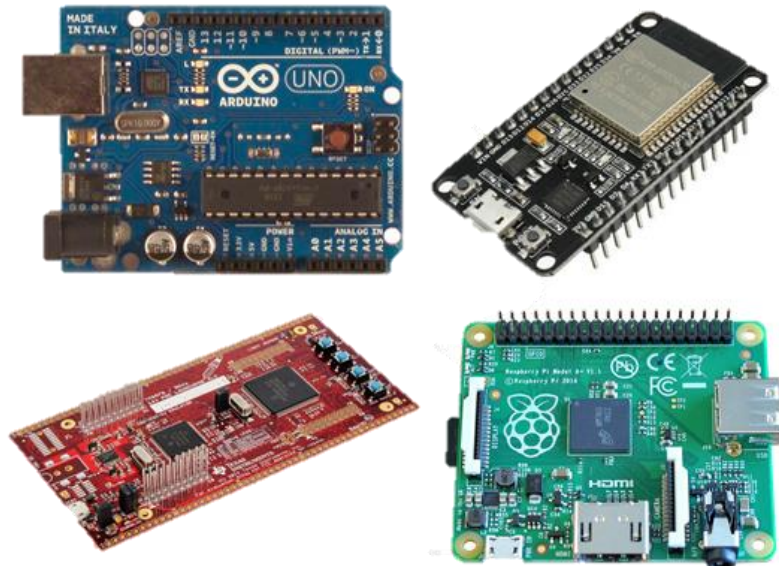


Figura 2.12 Tarjetas de microcontroladores.

2.3. Cinemática y rotaciones

Para el desarrollo del controlador es necesario tener la capacidad de conocer la posición del órgano efector de un robot mediante el conocimiento de los ángulos/desplazamientos de las articulaciones, o inversamente, poder determinar los ángulos/desplazamientos

necesarios de las articulaciones para ubicar el órgano efector en un punto concreto, esta información la da la cinemática del robot, por lo cual se expondrán algunos apartados a destacar de esta área.

2.3.1. Grado de libertad

Un objeto libre en el espacio tiene la capacidad para moverse en tres direcciones independientes y mutuas además de poder rotar en tres planos. Estos son los seis grados de libertad que puede tener un cuerpo, y que, trasladado a un robot se asemejaría al de la Figura 2.14, pero realmente el número real de grados de libertad que podrá poseer un cuerpo dependerá de su capacidad de movimiento dado por la cantidad de articulaciones que posee y las restricciones de estas que variaran según su diseño [8].



Figura 2.13 Ejemplificación de los grados de libertad en un robot

2.3.2. Modelo cinemático

El término cinemática (kinematic en inglés) es usado para el estudio del movimiento sin involucrar a las fuerzas. Cuando se consideran únicamente los movimientos sin las fuerzas o la energía involucrada entonces se habla de un análisis cinemático del mecanismo [8].

El movimiento de un cuerpo rígido se compone como una combinación de un movimiento de traslación y rotación. Al considerar las tres dimensiones del espacio, un movimiento de traslación es el movimiento generado a lo largo de uno o más de los tres ejes, como se ve

en la Figura 2.15 (a). Un movimiento rotacional puede ser considerado como la rotación en uno o más de los tres ejes, Figura 2.15 (b).

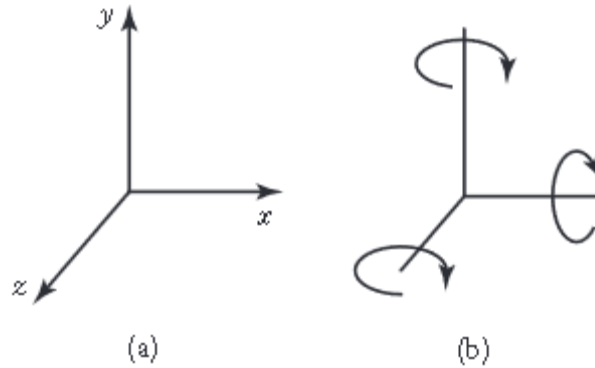


Figura 2.14 a) Movimiento traslacional b) Movimiento rotacional

El modelado cinemático se basa en la posibilidad de describir la posición de una herramienta y las localizaciones de cada uno de sus segmentos empleando un sistema común de coordenadas.

Típicamente los manipuladores traerán sensores que permitirán leer la posición en cada una de las uniones, con estas mediciones se pueden obtener los ángulos θ_1 y θ_2 en el caso de trabajar con un manipulador con dos uniones como el de la Figura 2.16. Lo que se necesitaría sería expresar las posiciones de los puntos finales en términos de los ángulos de dichas uniones [10].

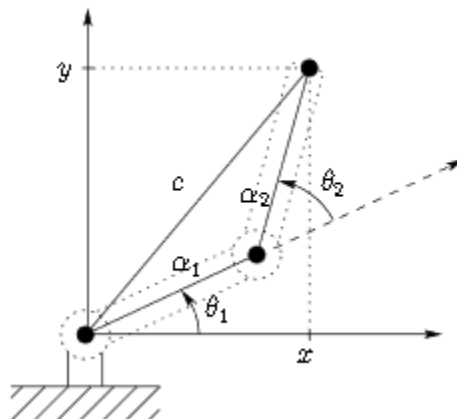


Figura 2.15 Ejemplo de una esquematización para el análisis cinemático

2.3.3. Cuaterniones

El campo de los números complejos forma un sistema numérico conveniente para trabajos en espacios bidimensionales, sin embargo, al tratarse de trabajos en espacios tridimensionales los números complejos se ven limitados.

Para trabajar un plano tridimensional con números complejos los usuarios tienden a ‘Eliminar’ una dimensión para llevar a cabo ciertas operaciones como traslación y rotación, para posteriormente reintegrarla solo para repetir el procedimiento hasta obtener el resultado deseado, trabajando únicamente mediante planos bidimensionales de un espacio tridimensional. El trabajar directamente en el espacio tridimensional ahorraría cálculos que se traduce en poder de procesamiento en un computador.

Dado este inconveniente es que se expone el uso de cuaterniones para el trabajo de modelación en el espacio, dado que los cuaterniones son expresados como un número real y tres números imaginarios, cada uno de ellos correspondiente a su propio conjunto de números $(a+bi+cj+dk)$ [7].

Tomando la definición de Hamilton, un quaternion Q está constituido por cuatro componentes (q_1, q_2, q_3, q_4) que representan las coordenadas del quaternion en una base $\{e, i, j, k\}$ [17]. Siendo común nombrar a la parte escalar ‘s’ del componente y del quaternion que es q_0 , y al resto como la parte vectorial (v) dejando a la representación tal que:

$$Q = [q_0, q_1, q_2, q_3] = [s, v] \quad 2.1$$

Para la utilización de los cuaterniones como metodología de representación de orientaciones se asocia el giro de un ángulo θ sobre el vector k al quaternion definido por:

$$Q = \text{Rot}(k, \theta) = (\cos(\theta/2), k \sin(\theta/2)) \quad 2.2$$

Donde:

$$k = u/||u|| \quad 2.3$$

De esta manera obtendríamos que la rotación expresada por un quaternion Q a un vector r sería definida por el producto:

$$Q \circ (0, r) \circ Q^* \quad 2.4$$

Donde:

$$Q^* = [q_0, -q_1, -q_2, -q_3] = [s, -v] \quad 2.5$$

Con ello es posible determinar las posiciones de las articulaciones de un robot [17].

Por lo que las operaciones relevantes para este proyecto son la adición y el producto de quaterniones.

Sea $H = \{(p_1, p_2, p_3, p_4) \mid p_1, p_2, p_3, p_4 \in \mathbb{R}\}$ el conjunto de los quaterniones, se definen sobre este, dos operaciones binarias de la siguiente manera. Sean $p = (p_1, p_2, p_3, p_4)$ y $q = (q_1, q_2, q_3, q_4)$ dos elementos del conjunto de los quaterniones H, entonces las operaciones binarias, basadas en el descubrimiento de Hamilton, quedan definidas como:

1. Una operación aditiva, $\oplus: H \rightarrow H$ definida de la siguiente manera

$$p \oplus q = (p_1 + q_1, p_2 + q_2, p_3 + q_3, p_4 + q_4) \quad \forall p \text{ y } q \in H \quad 2.6$$

2. Y una operación multiplicativa, $\otimes: H \rightarrow H$ definida mediante

$$\begin{aligned} p \otimes q &= (p_1, p_2, p_3, p_4) \otimes (q_1, q_2, q_3, q_4) \\ &= (p_1q_1 - p_2q_2 - p_3q_3 - p_4q_4, p_1q_2 + p_2q_1 + p_3q_4 - p_4q_3, \\ &\quad p_1q_3 - p_2q_4 + p_3q_1 + p_4q_2, p_1q_4 + p_2q_3 - p_3q_2 + p_4q_1) \\ &\quad \forall p \text{ y } q \in H \end{aligned} \quad 2.7$$

2.3.4. Arquitecturas

La posibilidad de colocar las uniones en un manipulador en cualquier parte permite la construcción de robots con distintas posibilidades de movimiento, aunque en la industria se emplean robots con arquitecturas estándares.

Existen dos tipos de uniones en los robots, la unión rotacional (R) que permite el movimiento de la articulación en ángulo, este se esquematiza como se ve en la Figura 2.17 a) y la unión prismática (P) que otorga movimiento lineal a la articulación, su esquema es el de la Figura 2.17 b).

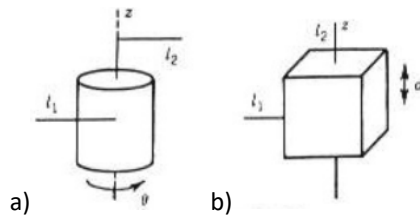

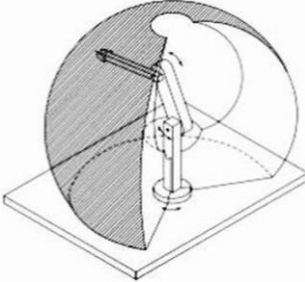
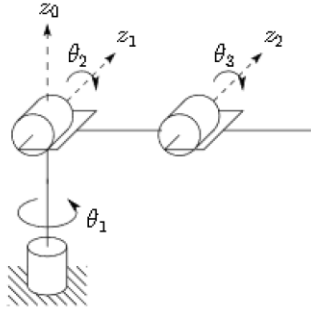

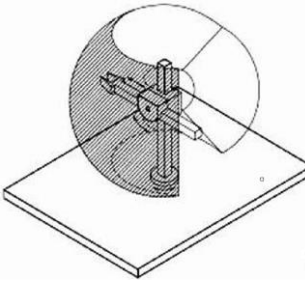
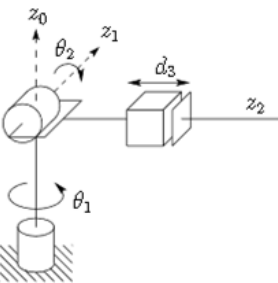

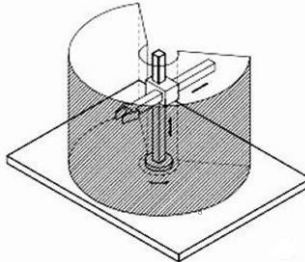
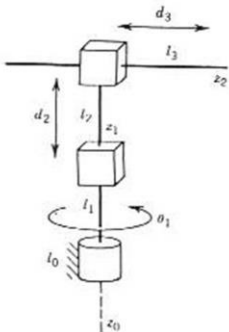

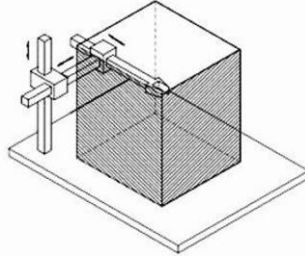
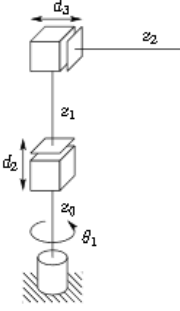


Figura 2.16 Tipos de articulaciones a) Rotacional, b) Prismática

Mediante la combinación y la posición de estos dos tipos de uniones se puede formar una variedad de robots, la Tabla 2.2 muestra robots con arquitecturas comunes en la industria [10].

Tabla 2.2 Arquitecturas comunes de robots industriales

Nombre	Ejemplo	Área de trabajo	Esquema
Manipulador revolvente o antropomórfico (RRR)	 Robot ABB IRB1400		
Manipulador esférico (RRP)	 Illinois, Stanford Arm		
Manipulador cilíndrico (RPP)	 PlateCrane EX		
Manipulador cartesiano (PPP)	 Epson Cartesian Robot		

2.3.7. Modelado cinemático CRS A465

Para la cinemática del CRS A465 se emplearán los procedimientos desarrollados por L. M. Juan Enrique Palomares Ruiz, en su trabajo de Modelación Cinemática Del Robot CRS A465 [7], cabe remarcar que se debería empezar por indicar las consideraciones que se tomaron en dicho trabajo como el nombramiento a las juntas y los giros del brazo robot dadas según la Figura 2.18.

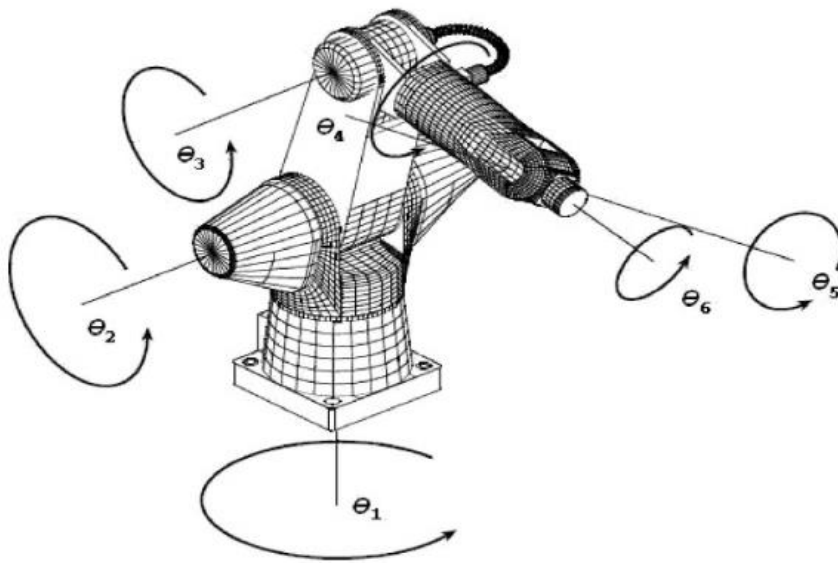


Figura 2.17 Localización de las juntas de rotación en el CRS A465

La rotación θ_1 , describe una rotación positiva a lo largo del eje de la z, en términos de la base inercial, que gira a todos los cuerpos. Las rotaciones θ_2 , θ_3 y θ_5 , se efectúan a lo largo del eje y en términos de la base inercial, estas rotaciones alejan al robot de la posición inicial. Las rotaciones θ_4 y θ_6 se realizan a lo largo del eje z, excepto cuando el robot se encuentra en una posición totalmente vertical, donde se presenta un fenómeno conocido en las ciencias computacionales como gimbal lock, ya que esta también puede categorizarse como una rotación en x, la descripción de estas rotaciones se puede observar en la Figura 2.19. Entonces cuatro de las rotaciones posicionaran el órgano efector en el espacio, para ser precisos θ_1 , θ_2 , θ_3 y θ_5 , donde a su vez θ_5 también sirve para orientar el órgano efector.

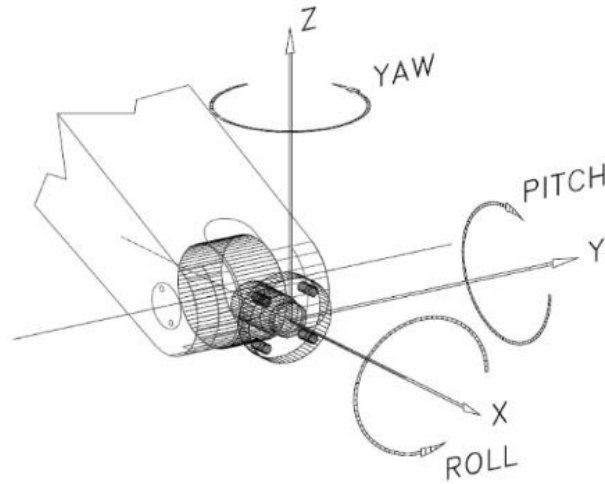


Figura 2.18 Localización de los últimos tres grados de libertad del CRS A465, conocidos como yaw, pitch & roll

Una vez dada la arquitectura del robot manipulador, se prosigue a declarar las bases y cuál será su orientación. En el presente trabajo se tomarán las bases como se muestra, en la Figura 2.20, pero la dirección de los giros será la opuesta a la que se muestra en dicha Figura, se debe recordar que para la simulación de la cinemática utilizando el álgebra de cuaterniones no es necesario seguir una convención en particular. Solo hay que definir las bases y el eje de la rotación.

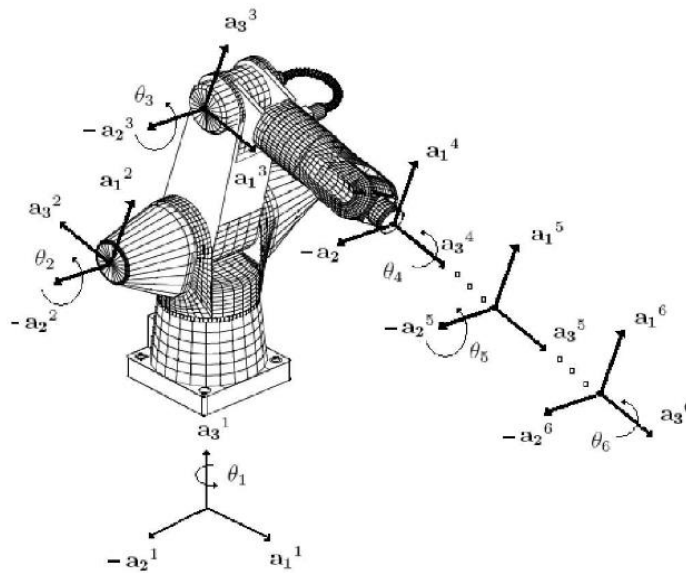


Figura 2.19 Declaración de las bases utilizando la regla de la mano derecha

CAPITULO III

METODOLOGÍA

En este capítulo se expone el proceso realizado para llevar a cabo el controlador.

3.1 Descripción del sistema a desarrollar

Debido a que no se posee información de un procedimiento general o que pueda ser empleado para este trabajo se ideó una metodología general para el desarrollo de controladores, esta es la que se presenta en el diagrama de flujo de la Figura 3.1.

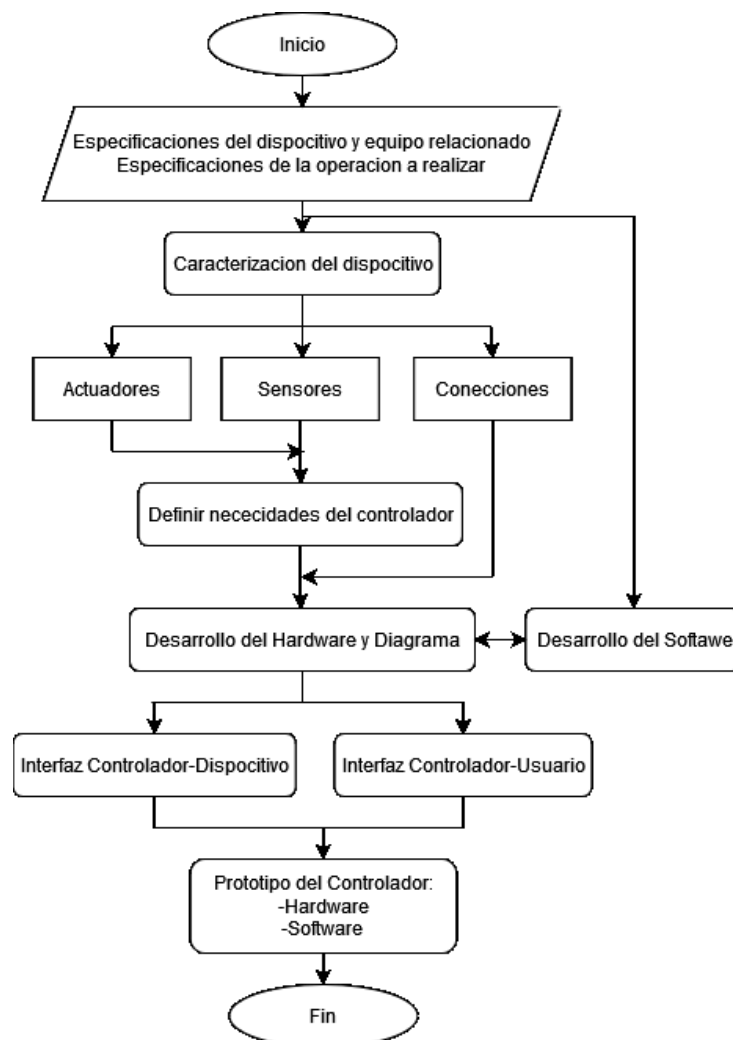


Figura 3.1 Metodología para el desarrollo de un controlador

Siguiendo la metodología la solución propuesta consiste en emplear un Arduino Due el cual sirva de intermediario entre el CRS A465 y un computador que tendrá una interfaz de usuario programada en Java, esto para poder comunicar al operador del robot con el microcontrolador, las relaciones entre dispositivos vienen como lo muestra la Figura 3.2.



Figura 3.2 Solución propuesta

Según se presenta en la Figura 3.2, el CRS A465 tiene una conexión directa de entradas y salidas con el Arduino Due, en esta relación el Arduino Due se encarga del posicionamiento de cada articulación.

El Arduino Due también cuenta con una conexión serial con una computadora que ejecutará la aplicación de Java, en esta conexión se transmitirán las instrucciones que provengan del computador y el estado actual del A465 proveniente del Arduino Due. La aplicación Java cuenta con una interfaz de usuario que ilustra la información del estado actual del robot y permite la inserción de comandos.

3.2 Materiales y equipo

El material y equipo empleados para llevar a cabo el proyecto se pueden dividir en el material físico o hardware que consiste en:

- 1) Brazo Robot CRS A465, esto compone la carcasa del robot, los motores, los encoders y complementos del propio robot como engranaje y tornillería.
- 2) C500C que es usado como carcasa para la protección del controlador.
- 3) Fuente de poder de +5v y 2.5A, fuente requerida para los encoders y puentes H.
- 4) Fuente de poder de +24v y 30A, empleada para energizar los motores.
- 5) Fuente de poder de +12v y 0.5A, empleada para energizar la pinza.

- 6) Arduino Due, la placa microcontrolador encargado del control del brazo.
- 7) Puentes H DC Mosfet IRF3205, uno para cada dos motores siendo un total de 3.
- 8) Tarjeta de relevador con optoacoplador, encargada del switcheo en la pinza.
- 9) Computador, donde será instalada la aplicación para el control del robot.

Y los programas o software donde se desarrollarán las interfaces:

- 1) Arduino, en el cual se realizará la programación del microcontrolador.
- 2) NetBeans IDE 8.2 y complementos, en el cual se realizará la programación de la interfaz del computador.

3.2.1. Caracterización del robot

El brazo A465 es un brazo robot con una pinza como órgano efector. Las uniones en las articulaciones permiten un movimiento de seis grados de libertad en el plano cartesiano, esto entrega un diseño del robot como el que se ve en la Figura 3.3 [18].

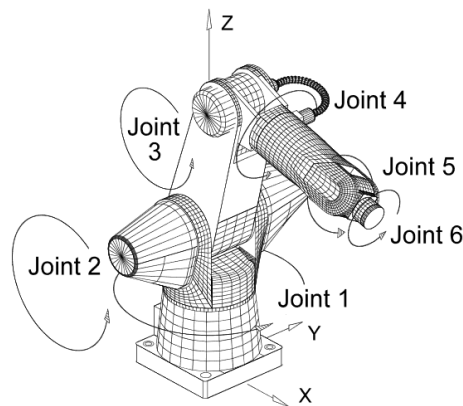


Figura 3.3 Brazo robot articulado A465

*Para el trabajo de este proyecto se considera una disminución en el ángulo de la articulación cuando dicha articulación gira en el sentido de la flecha mostrada en la Figura 3.3, este movimiento es inverso en el manual.

Las uniones del robot poseen unas limitaciones de movimiento dejando un rango teórico de valores permitidos para la rotación, además es necesario aclarar que para evitar daños al

mecanismo y componentes del robot se deben respetar ciertas velocidades y cargas máximas, en la Tabla 3.1 se describirán estas características del robot [19].

Tabla 3.1 Características del robot A465

Articulación	Rango de Movimiento	Par (Nm)	Velocidad Max.		Aceleración	
			(rad/s)	(°/s)	(rad/s ²)	(°/s ²)
J1 (Cintura)	+175° a -175°	39.50	3.14	180	12.6	720
J2 (Hombro)	+90° a -90°	66.08	3.14	180	12.6	720
J3 (Codo)	+110° a -110°	39.50	3.14	180	12.6	720
J4 (Muñeca rotatoria)	+180° a -180°	6.89	2.99	171	24.9	1430
J5 (Paso de muñeca)	+105° a -105°	6.82	3.02	173	25.1	1430
J6 (Herramienta)	+180° a -180°	2.50	2.99	171	24.9	1430
Peso Aproximado del Robot: 32Kg o 67lb						
Repetibilidad ± 0.05mm						
Capacidad de Carga	Máxima	80% de la velocidad o aceleración.			3.0 Kg	
	Normal	100% de la velocidad o aceleración.			2.0 Kg	

Además, se deberán considerar las dimensiones del robot dentro de un área de trabajo, siendo las posiciones que tiene capacidad de tomar dentro del espacio según sus articulaciones, esto se ilustra en la Figura 3.4.

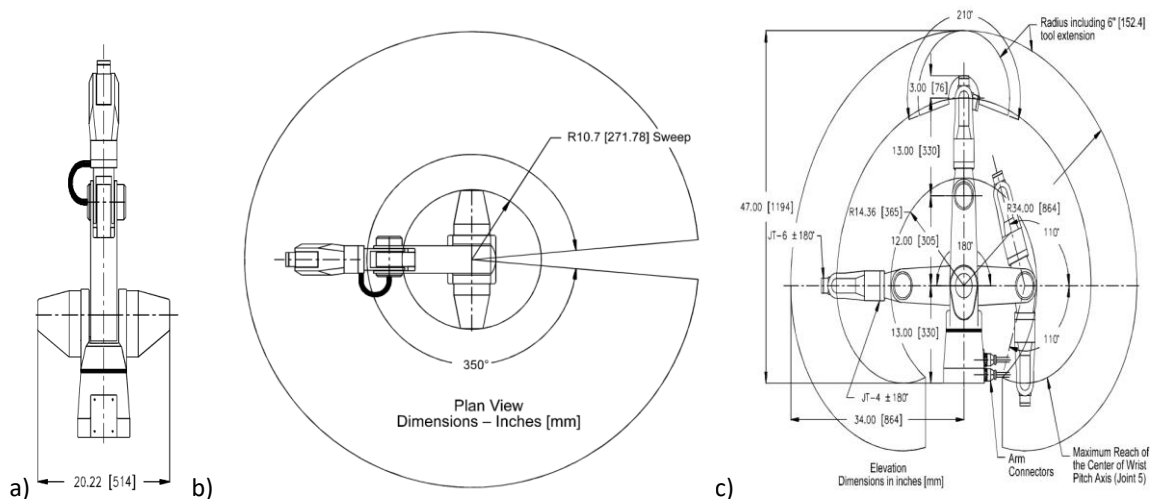


Figura 3.4 Rango de movimiento y dimensiones del A465 a) Ancho del robot b) Área de trabajo superior c) Área de trabajo lateral

En la Figura 3.5 los espacios máximos en los que se puede ubicar el órgano efector del brazo robot empleando una vista aérea.

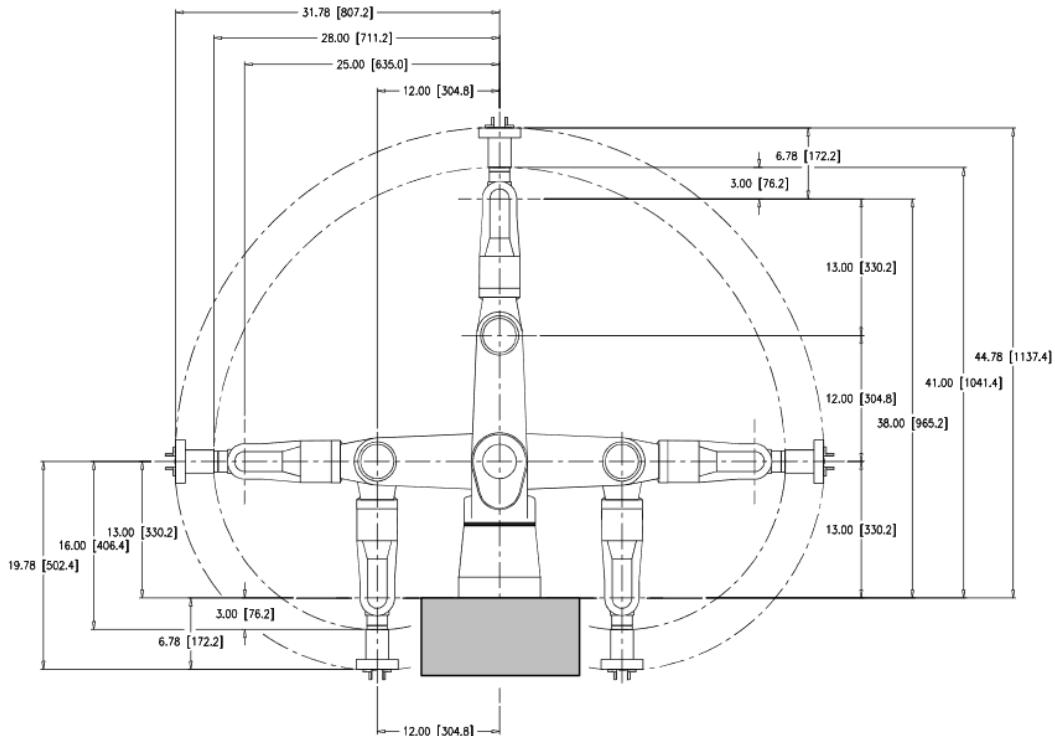


Figura 3.5 Rango de movimiento del efector final

3.2.2. Cálculo de la cinemática directa

Para obtener las coordenadas de la posición final junto con la orientación se empleó el uso de cálculos con cuaterniones, se emplearon los mismos métodos que los expuestos por en el trabajo de tesis *Modelación Cinemática Del Robot CRS A465, Utilizando El Álgebra De Cuaterniones*, pero invirtiendo el sentido de los giros del robot y añadiendo la distancia de la pinza.

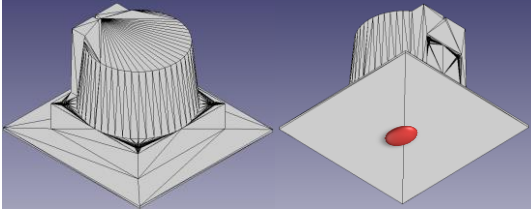
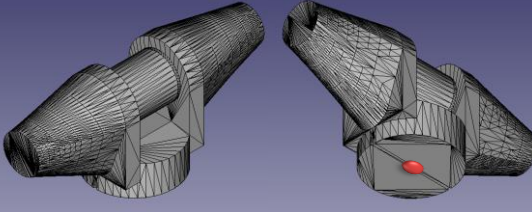
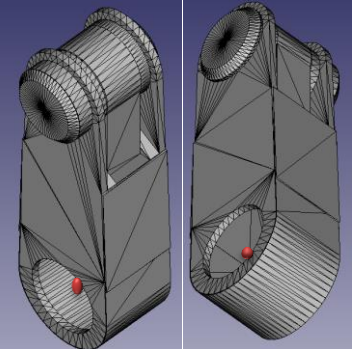
El procedimiento consistió de la siguiente manera:

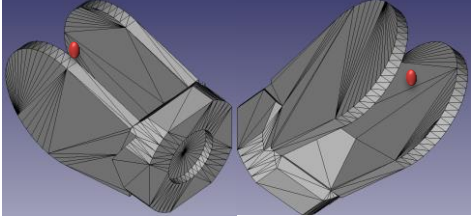
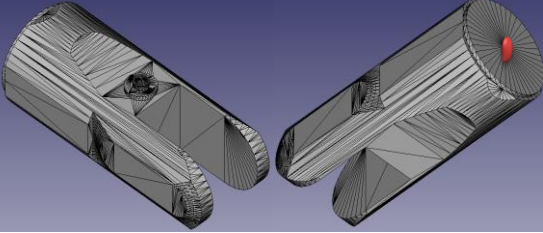
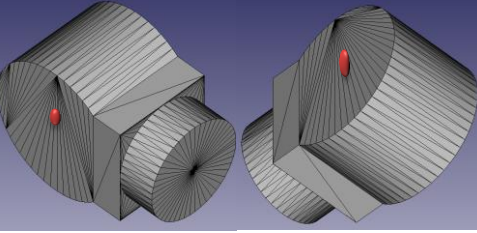
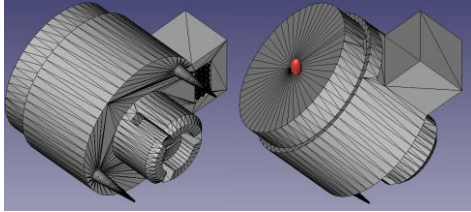
1. Se trató cada pieza del robot como un vector en el origen.

2. Se obtuvo la coordenada de la posición del vector con una rotación en su eje de rotación para ello se emplea la ecuación 2.4 para determinar el efecto del giro y la ecuación 2.6 para determinar la posición final.
3. En el caso de estar conectado a otra pieza que rotase se obtuvo una nueva coordenada de la posición del vector con una rotación esta vez en función del eje de la pieza inmediata anterior. Este paso se repite hasta llegar al inicio de la cadena.
4. Para los vectores de Roll, Pitch y Yaw se empleó la misma metodología, pero utilizando como vectores en el origen (1,0,0), (0,1,0) y (0,0,1).

Para los cálculos se usaron los puntos de la Tabla 3.2 más el tamaño de la pinza la cual se considero era de 5.5cm.

Tabla 3.2 Consideraciones para los cálculos

Pieza	Características
	<p>Nombre: Base Coordenada del punto: 0,0,0 Eje de rotación: NA</p>
	<p>Nombre: Hombro Coordenada del punto: 0,0,7 Eje de rotación: Z</p>
	<p>Nombre: Brazo Coordenada del punto: 0,0,13 Eje de rotación: X</p>

	<p>Nombre: Codo Coordenada del punto: 0,0,25 Eje de rotación: X</p>
	<p>Nombre: Antebrazo Coordenada del punto: 0,4,25 Eje de rotación: Y</p>
	<p>Nombre: Muñeca Coordenada del punto: 0,13,25 Eje de rotación: X</p>
	<p>Nombre: Mano Coordenada del punto: 0,16,25 Eje de rotación: Y</p>

Y se consideró el giro positivo de cada articulación como lo indican las flechas en la Figura 3.6.

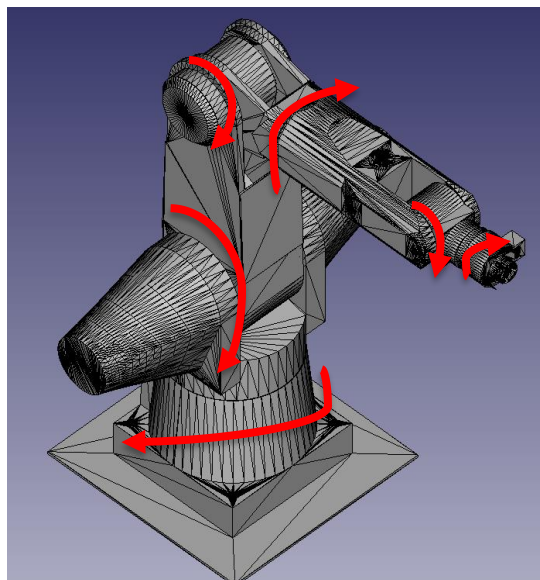


Figura 3.6 Consideraciones de los giros

Como posición inicial, donde los valores de los ángulos de todas las articulaciones son 0°, se decidió emplear la posición Home que se muestra en la Figura 3.7, esta relación no se empleara dentro de la interfaz del usuario debido a que la posición Home se consigue cuando las articulaciones poseen los valores de 175°,90°,200°,180°,105°,180° avanzando de la base a la pinza, por lo que serán ajustados a posterior.

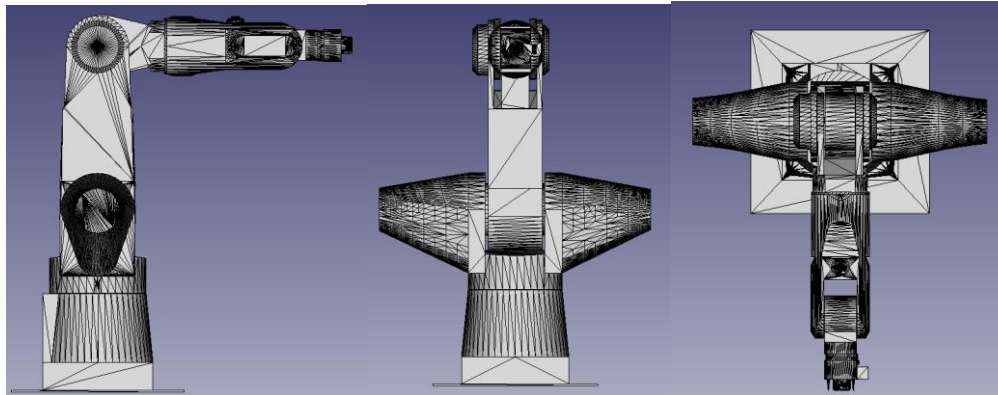


Figura 3.7 Consideraciones de la posición Inicial

Se realizaron los cálculos en el software de Wolfram Mathematica, en él se establecieron las operaciones y valores que dieron como resultado las siguientes ecuaciones.

Las coordenadas para la posición de la pinza fueron las siguientes:

$$X= 11. \cos\left[\frac{\theta_1}{2}\right]\cos[\theta_2]\cos[\theta_3]\cos[\theta_5]\sin\left[\frac{\theta_1}{2}\right] + 13. \cos[\theta_2 + \theta_3]\sin[\theta_1] + 12. \sin[\theta_1]\sin[\theta_2] - 8.5\cos[\theta_5]\sin[\theta_1]\sin[\theta_2]\sin[\theta_3] - 8.5\cos[\theta_3]\cos[\theta_4]\sin[\theta_1]\sin[\theta_2]\sin[\theta_5] + 5.5\cos\left[\frac{\theta_1}{2}\right]^2\sin[\theta_4]\sin[\theta_5] + 3. \cos[\theta_1]\sin[\theta_4]\sin[\theta_5] - 5.5\sin\left[\frac{\theta_1}{2}\right]^2\sin[\theta_4]\sin[\theta_5] + \cos[\theta_2]\sin[\theta_1](3. \cos[\theta_3]\cos[\theta_5] - 8.5\cos[\theta_4]\sin[\theta_3]\sin[\theta_5])$$

$$Y= \cos[\theta_1](13. \cos[\theta_2 + \theta_3] + 8.5\cos[\theta_2]\cos[\theta_3]\cos[\theta_5] + 12. \sin[\theta_2]) + 8.5\cos[\theta_5]\sin\left[\frac{\theta_1}{2}\right]^2\sin[\theta_2]\sin[\theta_3] + 8.5\cos[\theta_3]\cos[\theta_4]\sin\left[\frac{\theta_1}{2}\right]^2\sin[\theta_2]\sin[\theta_5] + 8.5\cos[\theta_2]\cos[\theta_4]\sin\left[\frac{\theta_1}{2}\right]^2\sin[\theta_3]\sin[\theta_5] - 8.5\sin[\theta_1]\sin[\theta_4]\sin[\theta_5] + \cos\left[\frac{\theta_1}{2}\right]^2(-8.5\cos[\theta_5]\sin[\theta_2]\sin[\theta_3] - 8.5\cos[\theta_4]\sin[\theta_2 + \theta_3]\sin[\theta_5])$$

$$Z= 13. - 8.5\cos[\theta_3]\cos[\theta_5]\sin[\theta_2] - 13. \sin[\theta_2 + \theta_3] + 8.5\cos[\theta_4]\sin[\theta_2]\sin[\theta_3]\sin[\theta_5] + \cos[\theta_2](12. - 8.5\cos[\theta_5]\sin[\theta_3] - 8.5\cos[\theta_3]\cos[\theta_4]\sin[\theta_5])$$

Y para la orientación se tienen las siguientes coordenadas en Roll:

$$X = \cos\left[\frac{\theta_1}{2}\right]^2 \cos[\theta_4] \cos[\theta_6] + \sin[\theta_1] (\cos[\theta_2 + \theta_3] (13 + 3\cos[\theta_5]) + 12\sin[\theta_2]) + \cos[\theta_6] \sin[\theta_1] \sin[\theta_2 + \theta_3] \sin[\theta_4] + 3\cos[\theta_1] \sin[\theta_4] \sin[\theta_5] - \cos[\theta_1] \cos[\theta_5] \sin[\theta_4] \sin[\theta_6] + \cos[\theta_2 + \theta_3] \sin[\theta_1] \sin[\theta_5] \sin[\theta_6] + \cos[\theta_4] (-\cos[\theta_6] \sin\left[\frac{\theta_1}{2}\right]^2 + \sin[\theta_1] \sin[\theta_2 + \theta_3] (-3\sin[\theta_5] + \cos[\theta_5] \sin[\theta_6])))$$

$$Y = -3\cos[\theta_2 + \theta_3] \cos[\theta_5] \sin\left[\frac{\theta_1}{2}\right]^2 - \cos[\theta_4] \cos[\theta_6] \sin[\theta_1] - \cos[\theta_6] \sin\left[\frac{\theta_1}{2}\right]^2 \sin[\theta_2 + \theta_3] \sin[\theta_4] + 3\cos[\theta_4] \sin\left[\frac{\theta_1}{2}\right]^2 \sin[\theta_2 + \theta_3] \sin[\theta_5] - 3\sin[\theta_1] \sin[\theta_4] \sin[\theta_5] + \cos\left[\frac{\theta_1}{2}\right]^2 (3\cos[\theta_2 + \theta_3] \cos[\theta_5] + \sin[\theta_2 + \theta_3] (\cos[\theta_6] \sin[\theta_4] - 3\cos[\theta_4] \sin[\theta_5])) + \cos[\theta_5] \sin[\theta_1] \sin[\theta_4] \sin[\theta_6] + \cos[\theta_1] (12\sin[\theta_2] + \cos[\theta_4] \cos[\theta_5] \sin[\theta_2 + \theta_3] \sin[\theta_6] + \cos[\theta_2 + \theta_3] (13 + \sin[\theta_5] \sin[\theta_6])))$$

$$Z = 13 + 12\cos[\theta_2] + \cos[\theta_2 + \theta_3] (\cos[\theta_6] \sin[\theta_4] - 3\cos[\theta_4] \sin[\theta_5] + \cos[\theta_4] \cos[\theta_5] \sin[\theta_6]) - \sin[\theta_2 + \theta_3] (13 + 3\cos[\theta_5] + \sin[\theta_5] \sin[\theta_6])$$

Pitch:

$$X = \sin[\theta_1] (\cos[\theta_2 + \theta_3] (13 + 4\cos[\theta_5]) + 12\sin[\theta_2]) + 4(-\cos[\theta_4] \sin[\theta_1] \sin[\theta_2 + \theta_3] + \cos[\theta_1] \sin[\theta_4]) \sin[\theta_5]$$

$$Y = -4\sin[\theta_1] \sin[\theta_4] \sin[\theta_5] + \cos[\theta_1] (\cos[\theta_2 + \theta_3] (13 + 4\cos[\theta_5]) + 12\sin[\theta_2] - 4\cos[\theta_4] \sin[\theta_2 + \theta_3] \sin[\theta_5])$$

$$Z = 13 + 12\cos[\theta_2] - 13\sin[\theta_2 + \theta_3] - 2\sin[\theta_2 + \theta_3 - \theta_5] - 4\cos[\theta_2 + \theta_3] \cos[\theta_4] \sin[\theta_5] - 2\sin[\theta_2 + \theta_3 + \theta_5]$$

Yaw:

$$X = 13\cos[\theta_2 + \theta_3] \sin[\theta_1] + 12\sin[\theta_1] \sin[\theta_2] + \cos[\theta_3] \cos[\theta_4] \cos[\theta_5] \cos[\theta_6] \sin[\theta_1] \sin[\theta_2] - 3\cos[\theta_5] \sin[\theta_1] \sin[\theta_2] \sin[\theta_3] - \cos\left[\frac{\theta_1}{2}\right]^2 \cos[\theta_5] \cos[\theta_6] \sin[\theta_4] + \cos[\theta_5] \cos[\theta_6] \sin\left[\frac{\theta_1}{2}\right]^2 \sin[\theta_4] - 3\cos[\theta_3] \cos[\theta_4] \sin[\theta_1] \sin[\theta_2] \sin[\theta_5] - \cos[\theta_6] \sin[\theta_1] \sin[\theta_2] \sin[\theta_3] \sin[\theta_5] + 3\cos[\theta_1] \sin[\theta_4] \sin[\theta_5] + \cos[\theta_2] \sin[\theta_1] (\cos[\theta_4] \sin[\theta_3] (\cos[\theta_5] \cos[\theta_6] - 3\sin[\theta_5]) +$$

$$\text{Cos}[\theta 3](3\text{Cos}[\theta 5] + \text{Cos}[\theta 6]\text{Sin}[\theta 5])) - (\text{Cos}[\theta 1]\text{Cos}[\theta 4] + \text{Sin}[\theta 1]\text{Sin}[\theta 2 + \theta 3]\text{Sin}[\theta 4])\text{Sin}[\theta 6]$$

$$\begin{aligned} Y = & -3\text{Cos}[\theta 2 + \theta 3]\text{Cos}[\theta 5]\text{Sin}[\frac{\theta 1}{2}]^2 - \text{Cos}[\theta 4]\text{Cos}[\theta 5]\text{Cos}[\theta 6]\text{Sin}[\frac{\theta 1}{2}]^2\text{Sin}[\theta 2 + \theta 3] + \\ & \text{Cos}[\theta 5]\text{Cos}[\theta 6]\text{Sin}[\theta 1]\text{Sin}[\theta 4] - \text{Cos}[\theta 2 + \theta 3]\text{Cos}[\theta 6]\text{Sin}[\frac{\theta 1}{2}]^2\text{Sin}[\theta 5] + \\ & 3\text{Cos}[\theta 4]\text{Sin}[\frac{\theta 1}{2}]^2\text{Sin}[\theta 2 + \theta 3]\text{Sin}[\theta 5] - 3\text{Sin}[\theta 1]\text{Sin}[\theta 4]\text{Sin}[\theta 5] + \text{Cos}[\frac{\theta 1}{2}]^2(\text{Cos}[\theta 4]\text{Sin}[\theta 2 + \\ & \theta 3](\text{Cos}[\theta 5]\text{Cos}[\theta 6] - 3\text{Sin}[\theta 5]) + \text{Cos}[\theta 2 + \theta 3](3\text{Cos}[\theta 5] + \text{Cos}[\theta 6]\text{Sin}[\theta 5])) + \\ & \text{Cos}[\theta 4]\text{Sin}[\theta 1]\text{Sin}[\theta 6] + \text{Cos}[\theta 1](13\text{Cos}[\theta 2 + \theta 3] + 12\text{Sin}[\theta 2] - \text{Sin}[\theta 2 + \theta 3]\text{Sin}[\theta 4]\text{Sin}[\theta 6]) \end{aligned}$$

$$\begin{aligned} Z = & -13(-1 + \text{Sin}[\theta 2 + \theta 3]) - \text{Cos}[\theta 3]\text{Sin}[\theta 2](3\text{Cos}[\theta 5] + \text{Cos}[\theta 6]\text{Sin}[\theta 5]) + \\ & \text{Sin}[\theta 2]\text{Sin}[\theta 3](-\text{Cos}[\theta 4]\text{Cos}[\theta 5]\text{Cos}[\theta 6] + 3\text{Cos}[\theta 4]\text{Sin}[\theta 5] + \text{Sin}[\theta 4]\text{Sin}[\theta 6]) + \\ & \text{Cos}[\theta 2](12 - \text{Sin}[\theta 3](3\text{Cos}[\theta 5] + \text{Cos}[\theta 6]\text{Sin}[\theta 5]) + \text{Cos}[\theta 3](\text{Cos}[\theta 4]\text{Cos}[\theta 5]\text{Cos}[\theta 6] - \\ & 3\text{Cos}[\theta 4]\text{Sin}[\theta 5] - \text{Sin}[\theta 4]\text{Sin}[\theta 6])) \end{aligned}$$

En la posición inicial el final de la pinza se encuentra en la coordenada 0,21.5,25 y la orientación sería Roll en 1,0,0, Pitch en 0,1,0 y Yaw en 0,0,1, cabe mencionar que en el trabajo se emplean pulgadas para las medidas.

3.2.3. Caracterización de la fuente de poder

Se analizaron las fuentes que posee el controlador C500C, estas se ven al remover las dos placas superiores del controlador como lo muestra la Figura 3.8



Figura 3.8 Fuente de poder del C500C

El controlador posee dos fuentes distintas de las cuales se pueden obtener distintos voltajes de acuerdo a la conexión que se realiza, los posibles voltajes se muestran en la Tabla 3.3

Tabla 3.3 Fuente de poder del C500C

Fuente de los Capacitores				
Color	Voltaje 1	Voltaje 2	Voltaje 3	Voltaje 4
Negro	86.5v	-	-	-
Blanco		-	-	-
Rojo	47.5v	-	95v	-
Azul		47.5v		-
Amarillo	-			-
Fuente del Embobinado				
Naranja	12v	-	-	-
Azul		12v	-	-
Verde	-		12v	-
Amarillo	17v	-		-
Rojo		5.2v	6.6v	19v
Verde	-			
Azul	-			
Naranja	-	-		

Estas fuentes cubren las necesidades de energía del robot, pero para la fabricación del controlador se emplearán fuentes ajenas que puedan ser conseguidas comercialmente o fabricadas permitiendo su remplazo en caso de una falla o ausencia del C500C, quedando las fuentes organizadas como se ve en la Tabla 3.4.

Tabla 3.4 Fuentes A, B y C

	Voltaje (v)	Corriente (A)	Propósito
Fuente A	24	30	Alimentar los motores
Fuente B	5	2.5	Alimentar los Encoders y puentes H
Fuente C	12	0.5	Alimentar la válvula neumática

3.2.4. Caracterización de los conectores

Se tienen dos conectores en el A465, uno manda la información de los encoders y el otro recibe la alimentación para los motores.

El cable que recibe la alimentación de los motores se trata de un conector CPC-24, las indicaciones del manual [18] muestran los pines de conexión además de informar que los motores 1, 2 y 3 requieren $\pm 70V$ y 12A, y los motores 4, 5 y 6 exigen $\pm 30V$ y 3A, en el Anexo A se puede ver la tabla completa del manual.

Sin embargo, los valores experimentales muestran que los motores poseen distintos rangos de voltaje, los cuales se muestran en la Tabla 3.5

Tabla 3.5 Conexiones y voltajes de los encoders

Articulación	Conexión+	Conexión-	Voltaje mínimo	Voltaje máximo
1	1	2	2.8v	12v
2	3	4	11.75v	15v
3	5	6	13.17v	17v
4	8	9	3.75v	12v
5	10	11	6.55v	12v
6	12	13	3.3v	12v

El cable que envía la información de los encoders se trata de un conector CPC-57, las indicaciones del manual [18] muestran los pines de conexión además de informar que los encoders son RS-422 los cuales tienen una resolución aproximada de 1000pulsos/° y que es por este mismo cable por el cual se alimentan, además también indica la conexión de la electroválvula, en el Anexo B se puede ver la tabla completa del manual.

Las conexiones que se emplearan para los encoders son sus señales A y B, y para la electroválvula es la interrupción de energía, la Tabla 3.4 muestra las conexiones de dichos encoders con sus correspondientes pines de alimentación y de la electroválvula.

Tabla 3.6 Conexiones y voltajes de los motores

Articulación	Señal		Voltaje	Tierra
	A	B		
1	1	2	15	24
2	3	4	16	25
3	20	21	17	40
4	22	23	52 y 51	55 y 57
5	43	49	52 y 51	55 y 57
6	19	26	52 y 51	55 y 57
Electroválvula	-	-	53	54

3.2.5. Caracterización del puente H

El puente H DC Mosfet IRF3205 que se muestra en la Figura 3.9 permite un control de voltaje de 3-36v con una corriente de 10A (30A máximo en picos), este requiere de dos fuentes de alimentación aisladas, una de 5v que alimente la entrada del puente H y la otra de la cual se suministrará la energía.



Figura 3.9 Puente H DC Mosfet IRF3205

El puente H permite controlar dos motores con una misma placa para ello cuenta con dos pines por motor, el de Dirección energizará de forma positiva al enviarle una señal de 1 y

de forma negativa si la señal es 0, la intensidad se regula con el PWM con el cual al enviar 0v detendrá los motores.

3.2.6. Caracterización del microcontrolador

Dados los objetivos del proyecto se decidió optar por emplear una tarjeta de microcontrolador en lugar de emplear únicamente el microcontrolador o un PLC, esto debido que la tarjeta permitirá un desarrollo más óptimo gracias a las ventajas con las que cuentan estas placas.

También se contempla que la placa cumpla las prestaciones de ser lo mayor comercial posible y que cuente con las especificaciones necesarias para el desarrollo del proyecto, es por ello que se decidió emplear la placa Arduino Due que se muestra en la Figura 3.10.



Figura 3.10 Arduino Due.

La placa Arduino Due pertenece a las placas desarrolladas por Arduino y se puede obtener comercialmente o en su defecto construir, la principal necesidad es que la placa cuente con mínimo 25 pines, de los cuales 12 deben poder ser usados como Interrupciones, y 6 como PWM, por lo que por las especificaciones de la placa mostradas en la Tabla 3.7, la placa cuenta con lo necesario y varios pines adicionales en caso de necesitar. Tiene la carencia con lo que respecta a la potencia llegando hasta 3.3v, pero esto puede ser resuelto añadiendo algunos aditamentos [16].

Tabla 3.7 Especificaciones del Arduino Due

Microcontrolador	AT91SAM3X8E
Voltaje de operación	3.3V
Voltaje de entrada	7-12V
Pines de E/S Digitales	54
PWM	12
Pines de entrada Analógica	12
Pines de salida analógica	2
Interrupciones	54
Velocidad de reloj	84MHz
Corriente de operación	800mA

3.3 Desarrollo de Interface Arduino-Robot

El desarrollo entre el microcontrolador y el CRS contempla la circuitería junto con sus conexiones eléctricas y la programación del Arduino Due.

Esta sección se puede trabajar de manera individual al resto de elementos ya que los únicos dispositivos que intervienen son el microcontrolador y el robot los cuales se deben comunicar como se ve en la Figura 3.11, esto ya se integrara posteriormente con el resto de elementos.

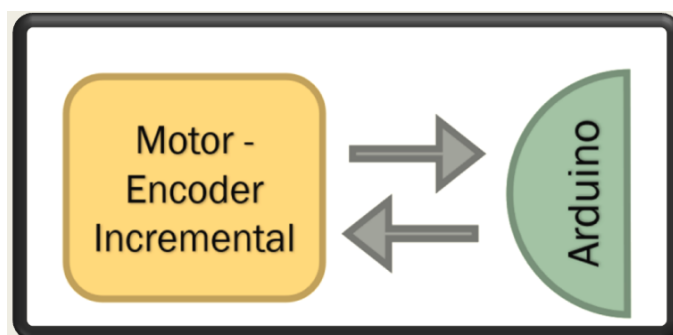


Figura 3.11 Motor-Encoder incremental con Arduino

3.3.1 Programación de la interfaz

La programación del Arduino consta de:

1. Obtener la posición de las articulaciones a partir de las lecturas de los encoders.
2. Mandar las señales de avance, retroceso y paro a los motores usando el puente H.
3. Desarrollar una función para mandar las articulaciones a su posición deseada.
4. Desarrollar una función para mandar las articulaciones a su posición de Home.

Para obtener la posición de las articulaciones se emplean las interrupciones del Arduino. Para que detecten los cambios en las salidas A y B de los encoders, dependiendo de la secuencia de la salida detectada se incrementará o decrementará un contador, y a continuación el conteo deberá ser transformado de su valor en pulsos a un valor de ángulo.

En cuanto a las señales que emite el microcontrolador hacia el puente H son tres:

- 1) Avance del motor: este hace que el motor se mueva en dirección opuesta al Home, considerando la configuración de los movimientos vista en la sección 3.2.1 Caracterización del robot, la función será utilizada cuando la posición deseada sea mayor que la posición actual.
- 2) Retroceso del motor: este hace que el motor se mueva en dirección al Home, considerando la configuración de los movimientos vista en la sección 3.2.1 Caracterización del robot, y será utilizada cuando la posición deseada sea menor que la posición actual.
- 3) Paro de motor: detener el movimiento del motor debe ser indicada cuando la posición actual y la posición deseada sean iguales o cuando llegue a Home.

El que una articulación llegue a una determinada posición requiere de dos valores base, la posición a la que desea llegar y la posición en la que se encuentra, a partir de ella se puede realizar una comparación para definir si la articulación debe girar en un sentido u otro, además un sistema de control para que, en medida de lo posible regule la intensidad del PWM teniendo en cuenta la inercia que generan los movimientos.

Un comando para enviar las articulaciones a su posición de Home consistiría en predefinir los ángulos de la posición Home como la posición deseada, y a partir de la posición actual hacer que, de forma individual, cada articulación avance o retroceda (dependiendo de su posición actual respecto a la de Home) hasta llegar a ella.

3.3.2 Conexiones

Cada articulación comprende dos conexiones a la placa de Arduino Due, una de ellas es conecta a la placa con el puente H y de este al motor, y la segunda es de la placa Arduino a los encoders.

En la Figura 3.12 se da una esquematización de las conexiones del Arduino Due con un puente H, en ella se ve como cada Puente H sirve para el control de dos motores (articulaciones), y que cada motor está conectado a un encoder el cual se conecta al Arduino para enviar la señal, adicional a lo mostrado el puente H se conecta por la parte de los motores a la Fuente A, y por la parte del microcontrolador se alimenta de la Fuente B al igual que los encoders, el Arduino es alimentado directamente del computador.

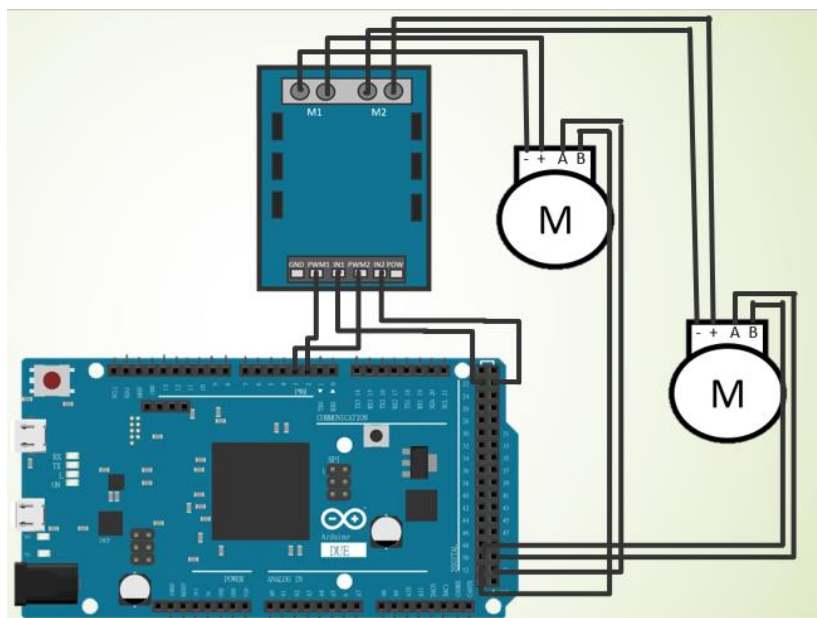


Figura 3.12 Esquematización de conexiones para dos motores

Para el control de la pinza se menciona en las indicaciones del manual (Anexo B) que esta funciona con una electroválvula de 12v, por lo cual se empleó un relevador para el control de la electroválvula conectado como se ve en la Figura 3.13; nuevamente el esquema es solo para esbozar las conexiones y no muestra la comunión de las tierras entre el relevador y el Arduino.

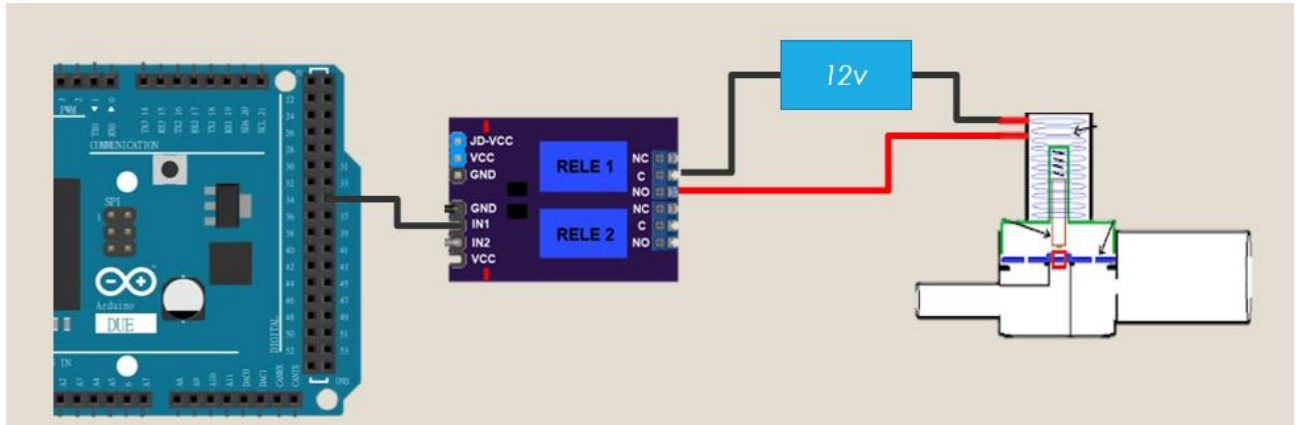


Figura 3.13 Esquematzación de conexiones para la electroválvula

Para realizar las conexiones se usó el adaptador del cable CPC-57 que trae el C500C el cual maneja la configuración de la Tabla 3.8.

Tabla 3.8 Conexiones de los encoders

-	-	-	-	47		-	-	42	35	34	44	37	36	45	46	39	38	30	29	41	10	9	14	6	5		
53	-	13	-	48		-	-	27	26	19	50	49	43	31	23	22	28	21	20	8	4	3	7	2	1		
G N D	55	55		23	23																						
	57	57	40	24	24																						
+	15	15	15	15	15																						
	16	16	16	16	16																						
	17	17	17	17	17																						
	50	50	50	50	50																						
	51	51	51	51	51																						

3.4 Interface Arduino-Java

La interacción entre el microcontrolador y el computador es por medio la comunicación de Serial una conexión USB.

Esta sección funciona como un intermediario entre la Interface de Arduino-Robot y la Interface de Java-Usuario ya que en ella se involucran el microcontrolador y el computador, Figura 3.11.

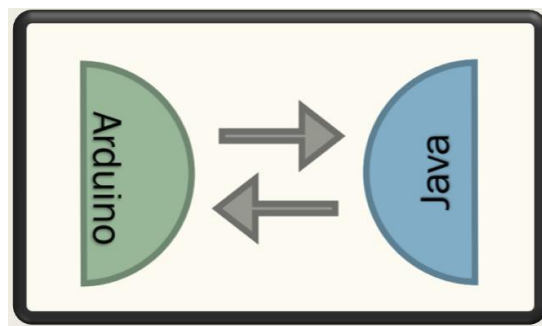


Figura 3.14 Arduino con Java

Dado que se pretende emplear el puerto serial para la comunicación es necesario que ambos programas tengan la capacidad para leer y escribir. Con respecto al Arduino este ya posee varias funciones que cumplen este propósito, sin embargo, este no es el caso con Java. Para permitir y facilitar la lectura y escritura en el puerto serial se emplearon las librerías de `jSerialComm-1.3.11.jar` y `arduino.jar`.

La comunicación de Java en el puerto serial no posee las mismas capacidades que Arduino, mientras Arduino es capaz de escribir y leer varios caracteres, Java únicamente puede escribir un carácter a la vez, aunque si es capaz de leer varios caracteres, por lo que se debe buscar la forma de superar estas limitaciones o en su defecto trabajar con ellas.

La Interface Arduino-Java debe enfocarse en los siguientes puntos:

1. Poder confirmar que existe conexión entre ambos.
2. Que Java pueda indicar al Arduino que ejecute alguna acción.
3. Que se pueda enviar una posición desde Java a Arduino.

3.5 Interface Java-Uusuario

Para el manejo del robot se contempla el uso de una interfaz gráfica creada en Java en la que el usuario pueda controlar el movimiento del robot mediante comandos y posea otras utilidades.

Esta sección trabaja con ventanas independientes a la comunicación con el Arduino, estas solo interactuaran con las instrucciones de Java en la clase principal en momentos puntuales, completando el desarrollo del sistema, Figura 3.15.

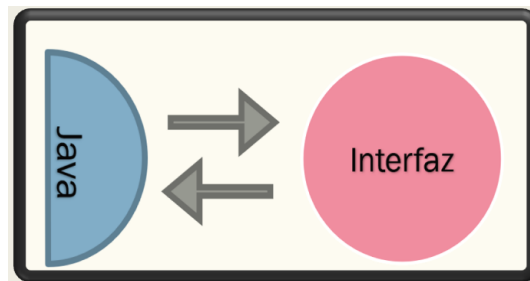


Figura 3.15 Interfaz con Java

Para el diseño de la interface se toma como inspiración la interface de COSIMIR que se muestra en la Figura 3.16, al observarla esta maneja una ventana principal a la cual se le sobreponen cuatro ventanas secundarias.

- Una simulación grafica tridimensional del Robot.
- Una tabla con posiciones almacenadas.
- Un espacio para escribir comandos.
- Una ventana de mensajes.

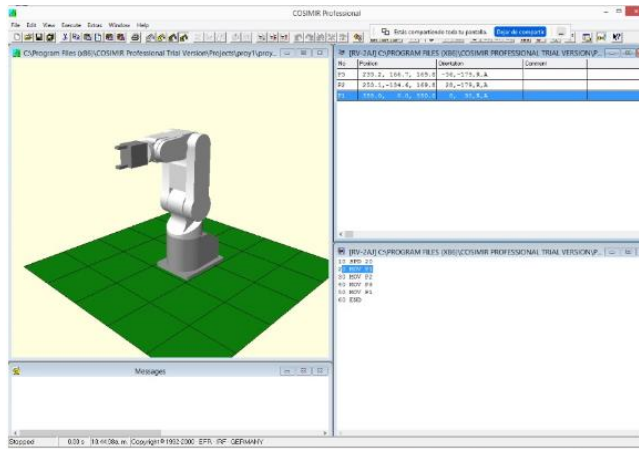


Figura 3.16 Interface COSIMIR

Teniendo en cuenta la Interface de COSIMIR se propusieron varios puntos para cada ventana.

En el caso de la ventana que presentará la simulación grafica 3D esta deberá poder mostrar la posición de las articulaciones del robot y su movimiento en tiempo real lo que brinde al usuario una vista de cómo es la trayectoria que sigue el robot al tratar de alcanzar alguna posición.

Para la ventana de comandos deberá poder almacenar una serie de instrucciones que el programa ejecute de manera ordenada.

En la ventana de la tabla que muestra las posiciones guardadas se tiene que entre la información que almacenará será el número referente a la posición almacenada, los ángulos en los que se encuentra cada articulación y la posición en el espacio que posee.

Finalmente se deberá tener una ventana que proporcione mensajes al usuario sobre el estado actual de la ejecución de los comandos ingresados y que muestre si se ha detectado algún problema durante la ejecución.

CAPITULO IV

RESULTADOS

En este capítulo se nombrarán los resultados obtenidos tras llevar a cabo la metodología del capítulo anterior.

4.1 Cinemática del robot

Las ecuaciones de la cinemática del Robot emplean como posición de origen, la posición que dentro de la interface es Home, por lo cual internamente de Java se realiza el ajuste restando los valores de los ángulos de la posición Home a la posición que entregan las ecuaciones. Tomando esto en consideración y utilizando el entorno gráfico de Wolfram Mathematica es posible comparar algunas configuraciones que puede tomar el robot, estas son mostradas en las Figuras 4.1, 4.2 y 4.3.

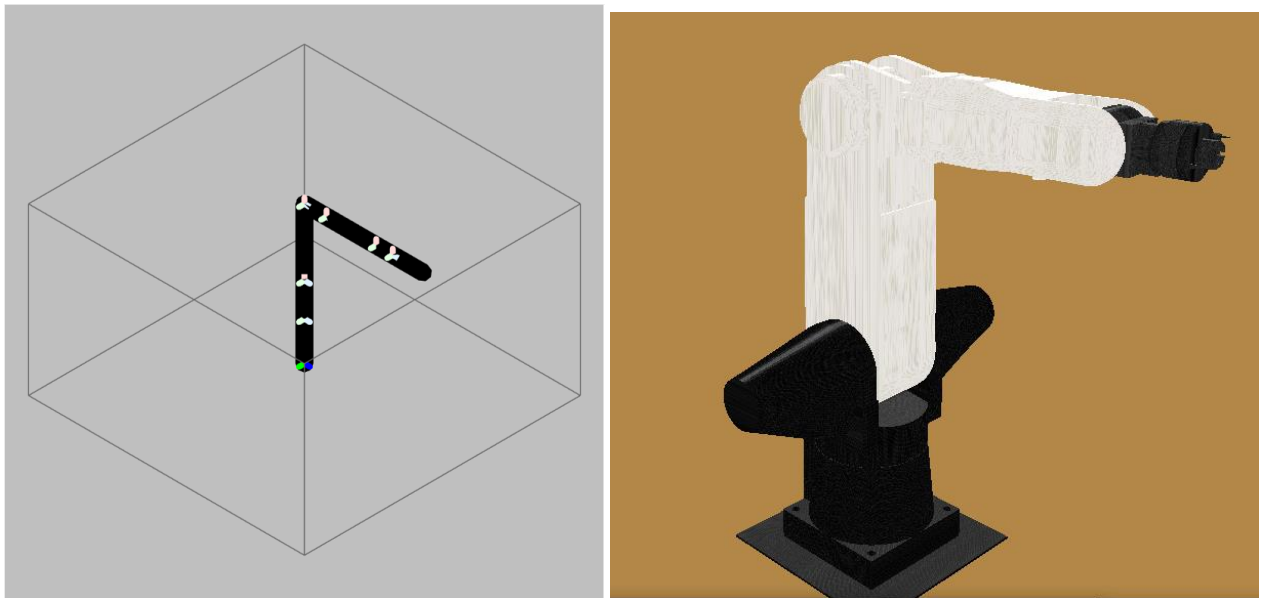


Figura 4.1 Configuración Home ($175^\circ, 90^\circ, 200^\circ, 180^\circ, 105^\circ, 180^\circ$)

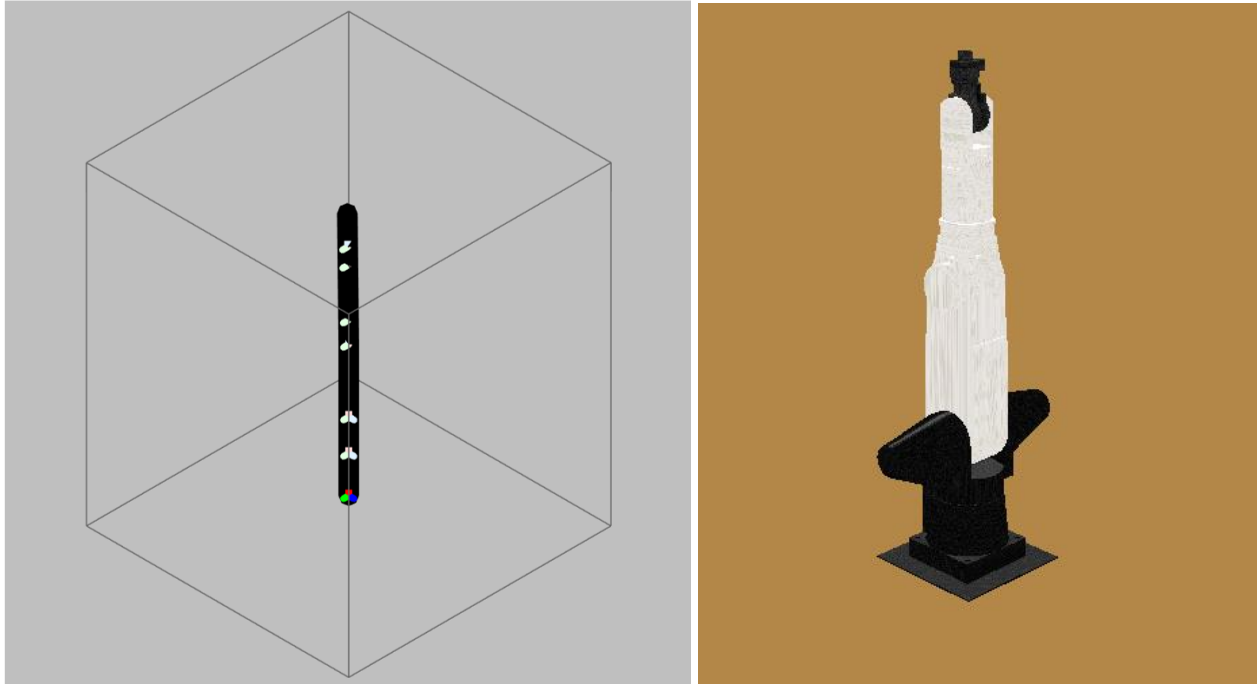


Figura 4.2 Posición Vertical ($175^\circ, 90^\circ, 110^\circ, 180^\circ, 105^\circ, 180^\circ$)

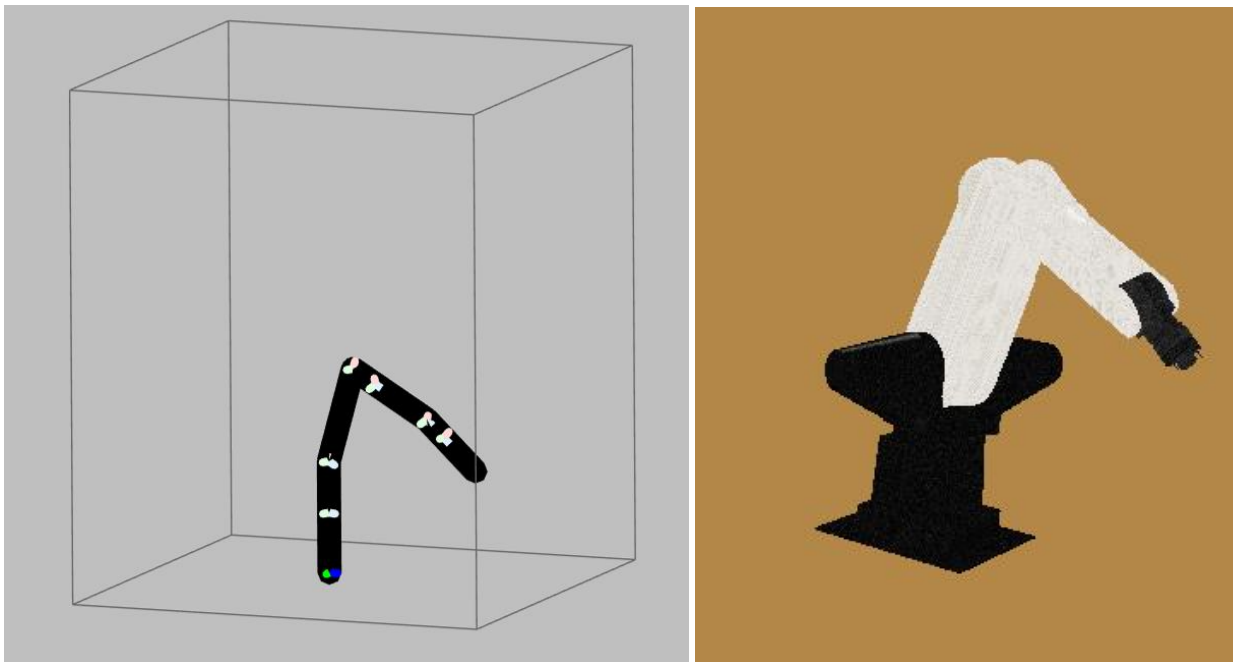


Figura 4.3 Configuración de muestra ($200^\circ, 110^\circ, 200^\circ, 150^\circ, 130^\circ, 220^\circ$)

Otra comparación puede ser realizada al considerar el trabajo de *Modelación Cinemática Del Robot CRS A465, Utilizando El Álgebra De Cuaterniones* [7] dado a que en él se emplean varias configuraciones, pero para hacer esto se deben contemplar que en el trabajo antes mencionado se considera una orientación distinta a la que se emplea en el

modelo, por mencionar un efecto de esto el robot extiende el órgano efector a lo largo del eje 'x' en el trabajo ya realizado mientras que en este se extiende a lo largo del eje 'y' ocasionando que estos valores se intercambien entre ambos resultados al igual que en los resultados de los cálculos de Roll, Pitch y Yaw, además de que en este trabajo se toma en consideración las dimensiones de la pinza por lo que adicional a la posición del punto de anclaje de la pinza también se tendrá el punto final de la misma y para terminar el cambio en de la posición de origen modifica el valor necesario de los ángulos para alcanzar un mismo punto. Así mismo la Tabla 4.1 presenta la comparativa de algunas configuraciones sin el ajuste de los ángulos al considerar Home.

Tabla 4.1 Comparativa de los modelos cinemáticos

Variables evaluadas	Resultados del modelo documentado	Resultados del modelo obtenido
Ángulos $\{\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6\}$ Pos (x,y,z) Anclaje (x,y,z) Roll (x,y,z) Pitch (x,y,z) Yaw (x,y,z)	{0°,0°,0°,0°,0°,0°} NA (0,0,41) (0,0,1) (0,1,0) (1,0,0)	{0°,0°,-90°,0°,0°,0°} (0,0,46.5) (0,0,41) (1,0,0) (0,0,1) (0,-1,0)
Ángulos $\{\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6\}$ Pos (x,y,z) Anclaje (x,y,z) Roll (x,y,z) Pitch (x,y,z) Yaw (x,y,z)	{0°,0°,-90°,0°,0°,0°} NA (16,0,25) (0,0,-1) (0,1,0) (1,0,0)	{0°,0°,0°,0°,0°,0°} (0,21.5,25) (0,16,25) (1,0,0) (0,1,0) (0,0,1)
Ángulos $\{\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6\}$ Pos (x,y,z) Anclaje (x,y,z) Roll (x,y,z) Pitch (x,y,z) Yaw (x,y,z)	{30°,40°,20°,0°,0°,0°} NA (-18.68, -10.7849, 30.1925) (-0.75, -0.433013, 0.5) (-0.5, 0.866025, 0) (0.433013, 0.25, 0.866025)	{30°,40°,20°,0°,0°,0°} (13.1665,22.805,32.9425) (10.7849,18.68,30.1925) (0.866025, -0.5, 0) (0.4330103, 0.75, 0.5) (-0.25, -0.433013, 0.866025)
Ángulos $\{\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6\}$ Pos (x,y,z) Anclaje (x,y,z)	{30°,40°,20°,10°,40°,0°} NA (-18.8085, -11.2458, 28.197)	{30°,40°,20°,10°,40°,0°} (14.4722,23.1691,27.2884) (11.2458,18.8085,28.197)

Roll (x,y,z)	(-0.221934, 0.0254671, 0.974729)	(0.809456, -0.567596, 0.150384)
Pitch (x,y,z)	(-0.567596, 0.809456, -0.150384)	(0.586627, 0.792831, -0.165191)
Yaw (x,y,z)	(-0.221934, 0.0254671, 0.974729)	(-0.0254671,0.221934,0.974729)
Ángulos $\{\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6\}$	{30°,40°,20°,10°,40°,10°}	{30°,40°,20°,10°,40°,10°}
Pos (x,y,z)	NA	(14.4722,23.1691,27.2884)
Anclaje (x,y,z)	(-18.8085, -11.2458, 28.197)	(11.2458,18.8085,28.197)
Roll (x,y,z)	(-0.792831, -0.586627, -0.165191)	(0.792737, -0.520434,0.317359)
Pitch (x,y,z)	(-0.520434, 0.792737, -0.317359)	(0.586627,0.792831,-0.165191)
Yaw (x,y,z)	(-0.317125, 0.165641, 0.933807)	(-0.165641,0.317125,0.933807)

Al analizar los valores en la tabla se observa que el cambio de orientación presenta una variación entre ambos modelos como es de esperar, a su vez interpretando los valores se observa como las posiciones correspondientes a los anclajes concuerdan en ambos casos dando fiabilidad al modelo obtenido, y como respaldo al observar la comparativa entre ambos entornos gráficos también se puede notar la similitud entre ambos modelos.

4.2 Interface Arduino-Robot

Debido a que el Arduino trabaja como intermediario entre el robot y el computador, éste posee instrucciones dedicadas al robot donde no interviene el computador.

4.2.1 Programación de la interfaz

Se establecieron doce instrucciones de interrupción en la programación del Arduino, dos por cada encoder, una por señal; estas son activadas al detectar la subida en un pulso en la señal lo cual permite detectar la dirección como lo muestra la Figura 4.4, en cada interrupción se tiene un comparador para limitar el tiempo entre lecturas, esto hace la función de un filtro pasa-bajas para señales de ruido, una vez pasado el filtro la señal sumara o restara valor a el contador de la articulación, esto es lo que se presenta en la sección del Anexo C.1.

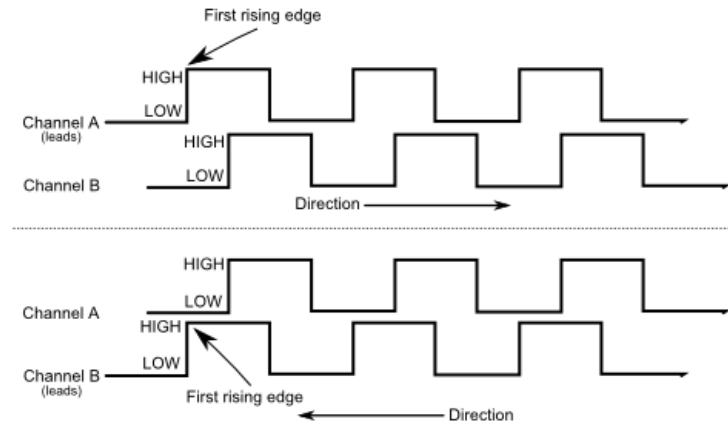


Figura 4.4 Pulsos e Interrupciones

Con un arreglo denominado `posicionActual[]` se almacena el conteo de lecturas que se tienen al momento y este arreglo es actualizado mediante la función que se ve en la sección del Anexo C.2.

Para ejecutar el avance, retroceso y paro del motor el Arduino deberá mandar dos señales al puente H, una que indique la dirección en formato binario y otra una señal que indique la intensidad en PWM:

- 1) Avance del motor: esta función solicita la intensidad y el motor que se desea activar, dentro de la función se manda la señal PWM y una señal de HIGH (3.3v) al puente H.
- 2) Retroceso del motor: esta función solicita la intensidad y el motor que se desea activar, dentro de la función se manda la señal PWM y una señal de LOW(0v) al puente H.
- 3) Paro de motor: esta función solicita el motor que se desea detener y dentro de la función se manda 0v como PWM al puente H.

Estas instrucciones son empleadas como funciones las cuales se presentan en la sección del Anexo C.3.

Contemplándolo como un diagrama de bloques aislado se tendría la Figura 4.5, donde las entradas que entran a la planta son las señales provenientes de cada encoder (Señal A y

B) y la posición deseada, y la salida sería cada uno de los motores de las articulaciones del robot.

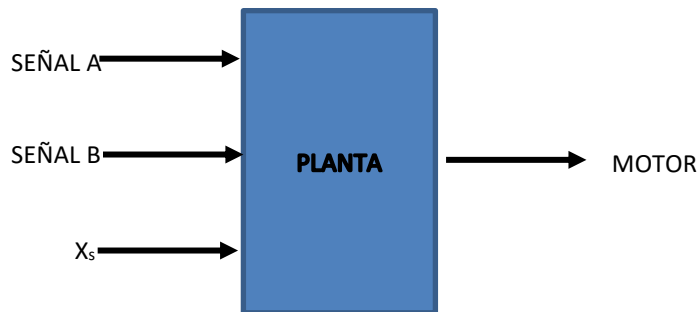


Figura 4.5 Diagrama de bloques

La función para modificar la posición actual de una articulación a una posición deseada se ve en el Anexo C.4, esta pide como parámetros el motor al que se desea manipular, el valor PWM del voltaje inicial (el voltaje mínimo mostrado en la tabla 3.5), el valor de lectura de pulsaciones deseado para la estabilidad y un tiempo de espera entre lecturas de comprobación.

Al comienzo de la función se reestablecen los contadores con su valor de posición actual almacenado, esto previene alguna lectura de ruido que haya podido distorsionar el valor de la posición actual.

Seguido entra en un bucle el cual se asegura de continuar el movimiento del motor hasta llegar a la posición deseada, el primer paso en el bucle es leer constantemente la nueva posición de la articulación y reestablecer los valores de las articulaciones ajenas a la que se pretende mover.

Lo siguiente es la comparación de la posición actual del motor con la posición deseada, en este caso hay tres posibles escenarios:

1. Ambas posiciones son iguales: En este caso el motor se detiene y se da por terminado el bucle.
2. La posición deseada es mayor a la actual: En este caso se usa la función para aumentar el ángulo.

3. La posición deseada es menor a la actual: En este caso se usa la posición para reducir el ángulo.

Para la medición del valor de lectura de pulsaciones deseado se comparan la posición actual de la articulación en dos tiempos distintos, este tiempo es definido por el valor de espera el cual es distinto para cada articulación y se encarga de regular la medida en la que se modifican las lecturas de los encoders mediante un cambio en la intensidad del PWM, esto muestra un efecto directo en el la velocidad de las articulaciones, sin embargo no se emplea ningún control de velocidad propiamente dicho, esto debido a que no se ha realizado un análisis dinámico involucrando las fuerzas de inercia, las fuerza de la gravedad, y demás factores que afectan a la velocidad y a que el desarrollo del prototipo tanto en Hardware como Software carece de los aditamentos para este fin, siendo como objetivo de esta función mantener el valor de lecturas de pulsaciones dentro de un rango.

Para el control del valor de lectura de pulsaciones se compara la medición de las lecturas en el tiempo y las lecturas ingresadas en el parámetro de la función y da como resultado dos situaciones:

1. Las lecturas de parámetro son mayores a la medida: Si esto ocurre se aumentará el pulso PWM para aumentar el voltaje en el motor, este aumento no supera un voltaje máximo.
2. Las lecturas de parámetro son menores a la medida: Si esto sucede se disminuye el pulso PWM para disminuir el voltaje en el motor, esta disminución nunca llega a 0v.

Adicional a esto las lecturas disminuirán cuando la articulación este llegando a la posición deseada.

Para el caso de mandar a Home las articulaciones consiste en girar la articulación hacia la posición Home (usando la función de mandar a una posición deseada), hasta que no se detecte un cambio en la posición actual, esta función sigue el diagrama de flujo de la Figura 4.6*.

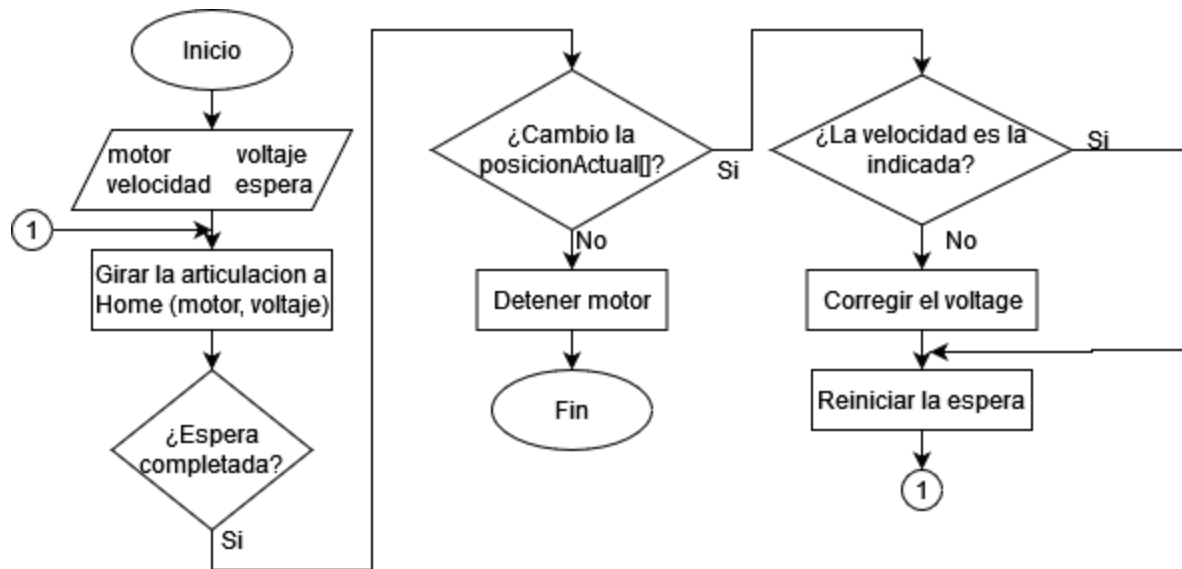


Figura 4.6 Función Home

*La variable velocidad hace referencia a las lecturas de pulsaciones deseadas

Todas estas funciones son resultado de aplicar la metodología y brindan un básico control del robot, aunque limitado en algunas áreas, a pesar de ello otorga un manejo suficiente para su uso contemplando ciertas restricciones.

4.2.2 Conexiones

Las conexiones se realizaron según las esquematizaciones mostradas en el apartado 3.3.2 Conexiones, estas comprenden las siguientes conexiones:

- 12 conexiones de los encoders al Arduino (un encoder por articulación y dos señales por encoder)
- 12 conexiones de los puentes H al Arduino (dos conectores por cada articulación, uno con el pulso PWM y otro con la señal de control)
- 1 conexión del relevador al Arduino
- 18 conexiones para alimentar la circuitería (2 conexión del Arduino para alimentar al relevador, 6 conexiones de la Fuente A para los motores, 8 conexiones de la Fuente B para los puentes H y el encoder y 2 conexión de la Fuente C para la electroválvula)

Una esquematización del diagrama eléctrico donde se presentan todos los elementos, que ya han sido descritos, como módulos independientes es mostrada en la Figura 4.7, y una versión ampliada puede ser encontrada en el Anexo E.

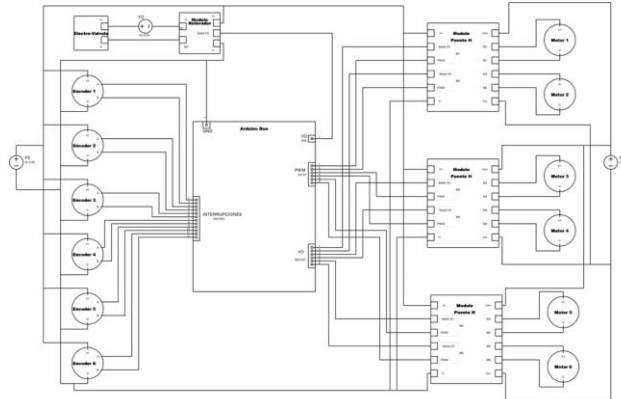


Figura 4.7 Diagrama eléctrico

La selección de los pines de las conexiones del Arduino Due con los componentes son las Mostradas en la Tabla 4.2.

Tabla 4.2 Distribución de los pines

PWM Arduino Due	Conexión	I/O Arduino Due	Conexión	Interrupción Arduino Due	Conexión
D2	Motor 1	D22	Motor 1	D42	Encoder 6 Señal A
D3	Motor 2	D23	Motor 2	D43	Encoder 6 Señal B
D4	Motor 3	D24	Motor 3	D44	Encoder 5 Señal A
D5	Motor 4	D25	Motor 4	D45	Encoder 5 Señal B
D6	Motor 5	D26	Motor 5	D46	Encoder 4 Señal A
D7	Motor 6	D27	Motor 6	D47	Encoder 4 Señal B
				D48	Encoder 3 Señal A
				D49	Encoder 3 Señal B
				D50	Encoder 2 Señal A
				D51	Encoder 2 Señal B
				D52	Encoder 1 Señal A
				D53	Encoder 1 Señal B

Todo fue ensamblado en una placa de madera y colocado dentro de la carcasa del C500C como lo muestra la Figura 4.8.

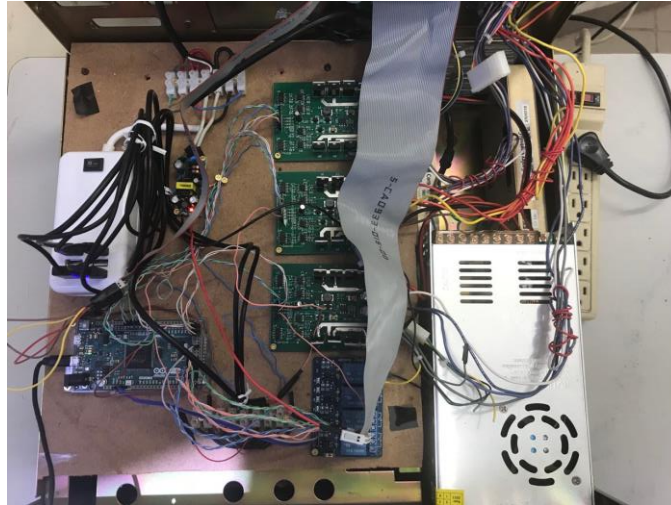


Figura 4.8 Conexiones del controlador

En comparativa con trabajos industriales el ensamblado presenta el problema de una posible desconexión o algún falso contacto entre dispositivos, siendo este el motivo por el cual se emplean placas con todos los componentes integrados, por su parte brinda la ventaja de dar un diseño modular para que en el caso de alguna avería o desperfecto cada parte pueda ser fácilmente reemplazada.

4.3 Interface Arduino-Java

La manera en que se trabajan con las limitaciones de la comunicación es empleando una metodología en la cual Arduino estará constantemente leyendo el puerto serial y Java enviará caracteres en momentos precisos, estos serán cuando se desee ejecutar alguna función de Arduino, Arduino poseerá varias funciones por defecto las cuales se ejecutará dependiendo del carácter recibido lo que da lugar a un código de caracteres, este es el que se muestra en el Anexo D.

En este código se debe mencionar que los caracteres correspondientes a los números (0,1,2,3,4,5,6,7,8,9) también se emplean, pero como los mismos números.

En el caso de una manera en que se confirme la comunicación entre Arduino y Java se tendría el código del Anexo C.5, en él se muestra como de manera cíclica mientras el puerto serial éste activo se leerá el valor que tenga, y este valor se comparará en una serie de casos dentro de un switch, para cuando se recibe el carácter 'a' el Arduino enviará un carácter '1' al puerto serial y encenderá el led de la placa.

Desde el punto de vista de Java se tiene el código del Anexo C.6, donde se creó una función para la conexión, enfocándose en la comunicación Java utiliza el objeto de la librería arduino.jar Arduino para escribir el carácter 'a' en el puerto serial, después lee el puerto serial para esperar la confirmación, y si el valor que captó corresponde a un '1' se indica que la conexión se estableció correctamente, en caso de recibir otro valor indica un fallo en la conexión.

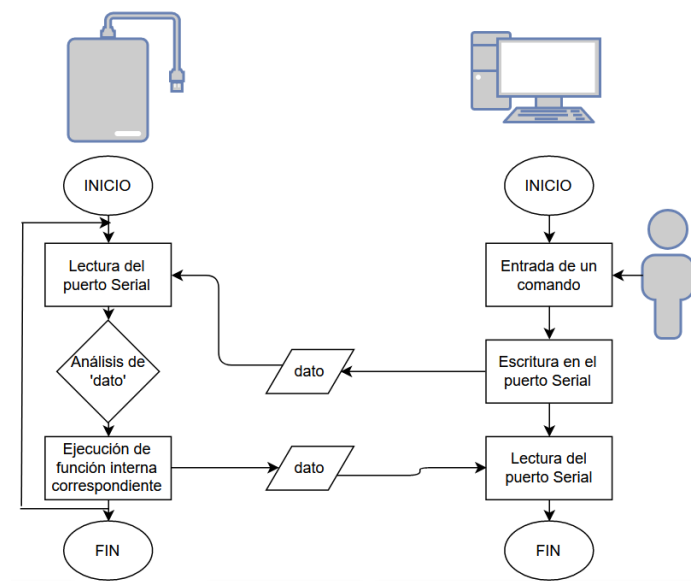


Figura 4.9 Comunicación Arduino Java

El método que usa Java para dar el resto de instrucciones al Arduino es similar, como se expresa en el código del Anexo C.7 primero manda un carácter al puerto serial, a continuación, monitorea en bucle en espera de un valor que confirme que el Arduino ha recibido y ejecutado la acción correspondiente y cuando Java recibe la confirmación puede llegar a tomar una acción o simplemente continuar con el resto de instrucciones, Figura 4.9.

La interacción más grande se lleva a cabo cuando Java debe mandar al Arduino que mueva a una nueva posición alguna articulación, para esta situación se creó un procedimiento específico

1. Mientras el Arduino está leyendo el puerto serial Java manda el carácter 'f' que indica al Arduino que java enviará una nueva posición deseada.
2. Java mandara el carácter 'j' el cual indica al Arduino que el siguiente carácter hace referencia a la articulación que tendrá la nueva posición.
3. Java manda un carácter del 0 al 5 que hace referencia a la articulación a mover.
4. Java envía el carácter 'c' el cual indica que el siguiente carácter hace referencia al valor de la centena del ángulo nuevo.
5. Java envía un carácter de 0 a 9 indicando el valor de la centena del nuevo ángulo.
6. Los pasos 4 y 5 se ejecutan nuevamente, pero enviando una 'd' para indicar la decena y 'u' la unidad del nuevo ángulo.
7. Finalmente, Java envía el carácter 'x' que indica que ya se ha enviado toda la información.

Visto desde Java la función es la que se muestra en el Anexo C.8, la función solicita la articulación a mover y el ángulo nuevo, entonces lo primero que hace es desglosar el ángulo en su valor de centena, decena y unidad y pasar los valores a caracteres, esto último se repite para el valor de la articulación.

En Arduino la función es la que se muestra en el Anexo C.9, en ella se preparan dos variables, la primera es un arreglo que almacenara los valores de la articulación a mover y la centena, decena y unidad del nuevo ángulo, y la segunda es un índice que permite moverse dentro del arreglo de la primer variable, a continuación entra en un bucle donde está constantemente leyendo el puerto serial y mediante un switch va almacenando los valores en el arreglo, cuando se ha leído la posición sale del bucle y reestructura el valor del ángulo y establece internamente la nueva posición.

Estas funciones fueron obtenidas con la metodología y permiten una comunicación entre el Arduino y Java, más sin embargo no hace a ambos dispositivos codependientes ya que

ambos equipos trabajan de manera separada escribiendo y leyendo en el puerto serial por lo que se facilita la modificación separada de cada uno de ellos.

Interface Java-Usuario

Para la interfaz gráfica se usó como primera pantalla una ventana en la cual se muestre algo de información sobre el proyecto y tres opciones principales, esta es la que se ve en la Figura 4.10.



Figura 4.10 Ventana principal

En ella cada botón posee una función en específico:

- **AYUDA:** Mostrara una ventana en la cual se dará la información básica para que un usuario pueda trabajar con el software al igual que indica sobre algunas precauciones y situaciones a tomar en cuenta cuando se manipula el programa, véase la Figura 4.11.
- **SIMULACION:** Permite trabajar con el programa en un modo desconectado al robot, esta opción permite al usuario trabajar con los comandos y la tabla de posiciones al mismo tiempo que visualiza todo desde la simulación grafica 3D del robot para un posterior uso con el robot.

- INICIAR: Dara inicio al protocolo para conectarse al controlador y manipular el robot, el ambiente será igual al que se obtiene en la opción de SIMULACION con la principal diferencia de que en esta ocasión la ejecución de los comandos será echa por el robot a la vez que se ejecuta la simulación.

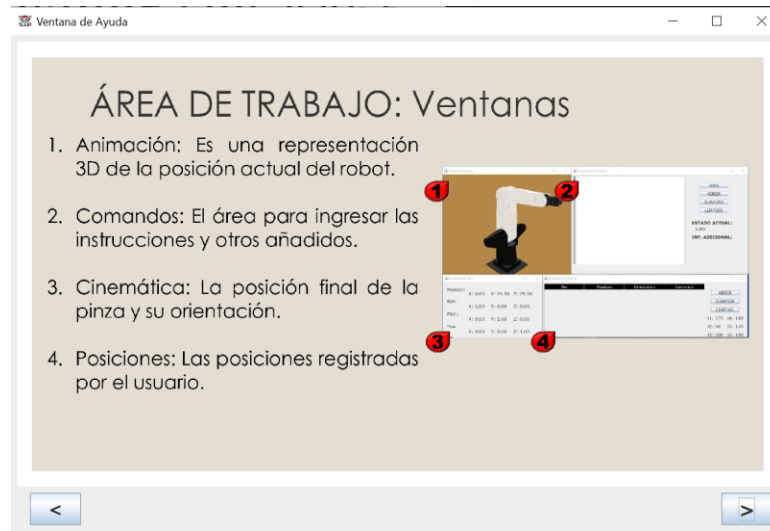


Figura 4.11 Ventana de Ayuda

La diferencia entre trabajar conectado al robot o solamente usar el simulador es el hecho de que se deberá completar la conexión al robot mediante la ventana de conexión que se ve en la Figura 4.12, en ella se solicita el puerto al cual se conectó el robot en la computadora y la velocidad de transmisión en baudios.

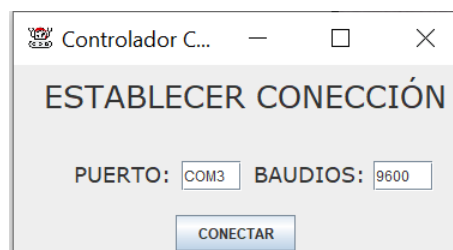


Figura 4.12 Ventana de conexiones

El área de trabajo se compone de cuatro ventanas como se ve en la Figura 4.11:

- Ventana de simulación: Presenta el A465 en un gráfico 3D el cual representa la posición de las articulaciones y muestra el movimiento de las mismas durante la ejecución de los comandos.
- Ventana de Comandos: Tiene un área para ingresar los comandos del usuario, éstos deben corresponder a los comandos programados, la ventana posee cuatro botones, RUN que permite ejecutar los comandos una vez ingresados, GUARDAR y ABRIR que permite salvar o abrir archivos de textos con comandos previamente programados y LIMPIAR que borra los comandos ingresados; adicional a esto en la misma ventana se muestran mensajes cuando está en ejecución algún programa.
- Ventana de Posiciones: Aquí se encuentra la tabla donde se almacenan las posiciones que ha guardado el usuario, la tabla muestra cuatro columnas, el número de referencia de la posición, los ángulos de las articulaciones, las coordenadas en el espacio y un área para escribir algún comentario; y tiene tres botones ABRIR y GUARDAR para cargar o salvar la tabla en un archivo txt y LIMPIAR lo cual borra las posiciones de la tabla; también posee un apartado donde muestra el último ángulo en que se dejaron las posiciones.
- Ventana de Cinemática: Esta ventana muestra las coordenadas en el espacio en la que se encuentra el final de la pinza, también las coordenadas de los ejes Roll, Pitch y Yaw para la orientación para ello se emplearon las ecuaciones del apartado 4.13 Cinemática del Robot.

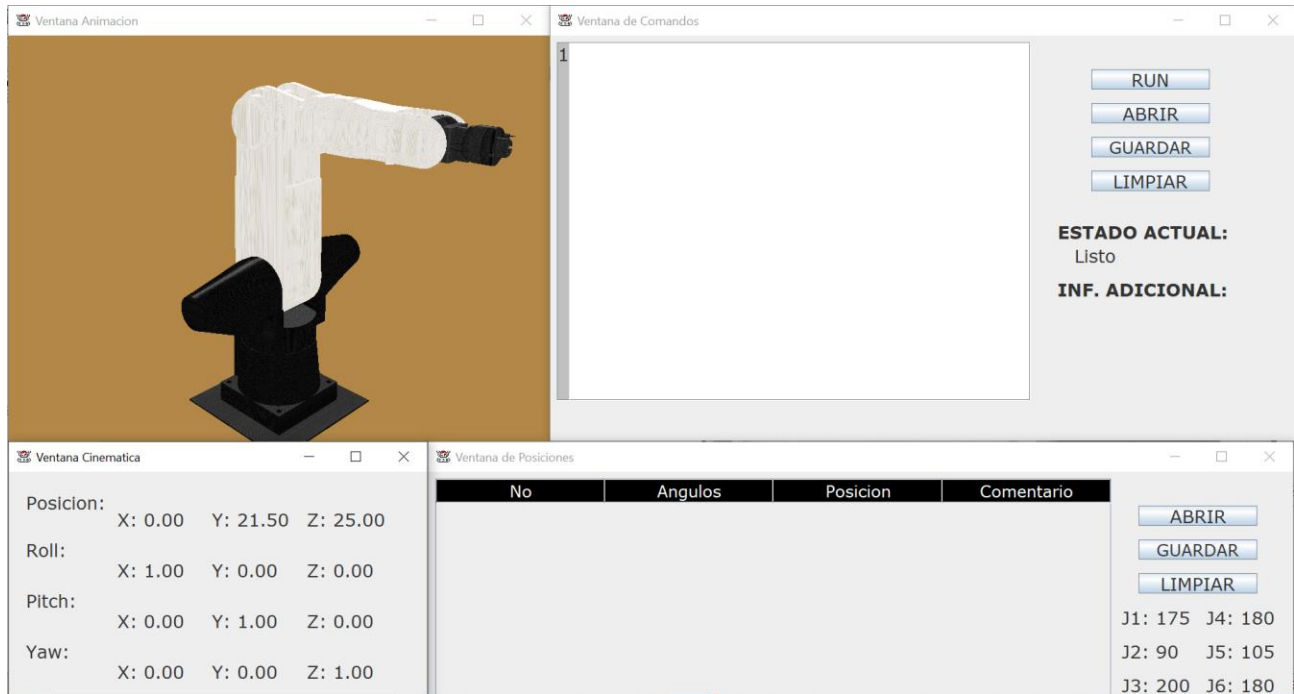


Figura 4.13 Área de Trabajo

Para el manejo del robot y de la interfaz se crearon 9 comandos que se muestran en la Tabla 4.3, de los cuales 4 solo pueden ser empleados si se realiza la conexión con el controlador del robot debido a que estos trabajan directamente con la tarjeta de microcontrolador, los otros 5 comandos pueden ser empleados en conexión con el robot o en la simulación.

Tabla 4.3 Comandos programados

Comando	Función
Home	Mueve las articulaciones a la posición de Home
Joint X,Ang	Mueve la articulación ' X ' al valor ' Ang '
move Ang1,Ang2,Ang3,Ang3,Ang4,Ang5,Ang6	Mueve todas las articulaciones a los ángulos indicados
here pX	Guarda la posición actual en la tabla de posiciones
move pX	Mueve las articulaciones a la indicada en la tabla de posiciones

clamp	Alterna entre abrir y cerrar la pinza
Adjust	Establece los ángulos de las articulaciones en 175,90,110,180,105,180
SetHome	Establece la posición actual como Home con los ángulos 175,90,200,180,105,180
values	Invoca una Ventana con los valores que posee el Arduino

4.4 Resultados experimentales

Tomando la información del controlador original (C500C) desarrollado por CRS que se presenta en los manuales [18][20] se comparan algunos aspectos con el nuevo controlador desarrollado en este proyecto.

Debido a las necesidades y límites del proyecto este posee algunas características distintas, la Tabla 4.4 muestra algunas.

Tabla 4.4 Comparación de características generales

Característica	Original	Nuevo
Angulo 0° en articulaciones	Cuando el robot está completamente vertical posee todas sus articulaciones en el ángulo 0°.	El límite de giro en sentido antihorario* de cada articulación es su ángulo 0°.
Sentido de giro	Antihorario*	Horario*
Velocidad	La velocidad es indicada desde la interfaz.	La velocidad no es medida, en cambio se plantean lecturas de pulsaciones constantes y variantes dependiendo de la articulación.
Interfaz de usuario	Ventana de comandos	Ventana de comandos Simulador Tabla de posiciones Trabajo con Archivos
Obtención	Ninguna (Anteriormente directa con el fabricante)	Online

*Tomando en cuenta el eje de la Tabla 3.2

La posición de Home, la cual se trata como la posición neutral del robot, es distinta debido a que se considera un distinto ángulo de origen en las articulaciones, este cambio se ve en la Tabla 4.5.

Tabla 4.5 Ángulos en la posición Home

Articulación	Original	Nuevo
Hombro	+175°	+175°
Brazo	+10°	+90°
Codo	-60°	+200°
Antebrazo	+180°	+180°
Muñeca	+25°	+105°
Mano	+180°	+180°

Se modificaron los rangos de movimientos del robot, en la Tabla 4.6 podemos ver los valores originales y nuevos con los que trabaja cada articulación y su desplazamiento total.

Tabla 4.6 Rangos de movimiento

Articulación	Original		Nuevo		Reducción
Hombro	+175° a -175°	350°	+340° a +10°	330°	20° (5.7%)
Brazo	+90° a -90°	180°	+180° a 0°	180°	0° (0%)
Codo	+110° a -110°	220°	+220° a +110°	110°	110° (50%)
Antebrazo	+180° a -180°	360°	+350° a +10°	340°	20° (5.6%)
Muñeca	+105° a -105°	210°	+200° a +10°	190°	20° (9.5%)
Mano	+180° a -180°	360°	+350° a 10°	340°	20° (5.6%)

El error del giro de cada articulación no viene expuesto en el manual (solamente indica un error de $\pm 0.05\text{mm}$) por lo que solo se tiene la del nuevo controlador, para esta prueba se manejó el un giro no mayor a 50° y con el resto de articulaciones (exceptuando a la que se estaba probando) estando en la posición de Home, la prueba se repitió 5 veces en cada articulación y el promedio del error es el que se muestra en la Tabla 4.7.

Tabla 4.7 Error en articulaciones

Articulación	Nuevo
Hombro	+4.6°
Brazo	+2.8°
Codo	+2.4°
Antebrazo	+3.2°
Muñeca	+1.6°
Mano	+2°

Finalmente existen una serie de fallos que no se pudieron abordar de los cuales no se pudo obtener su causa y que por lo tanto llegan a presentarse ocasionalmente, estos son los siguientes:

- Al establecer movimientos mayores de 5° o 10° en un solo comando la articulación sobrepasa el ángulo deseado.
- La posición a partir de la cual se parte al ejecutar un movimiento influye en su error.
- La ejecución de algunos comandos en cierto orden puede llegar a causar incongruencias con la representación gráfica.
- Dependiendo de los comandos y el equipo la simulación puede desfasarse del movimiento real del robot.
- En movimientos de 1° o 2° grados se presenta un error
- El cableado interno llega a moverse
- Articulaciones 4 y 5 no actúan según lo indicado comúnmente.

CAPITULO V

CONCLUSIONES

Las conclusiones obtenidas tras la realización del proyecto fueron las siguientes.

5.1 Conclusiones del proyecto

El objetivo de la presente tesis consistió en desarrollar un controlador para el brazo A465 un robot fabricado por CRS Robotics que posee 6 grados de libertad y que la institución guardaba desde hace varios años totalmente en desuso, tras el proyecto se logró darle nuevamente movilidad al equipo con el uso de la tecnología y conocimientos actuales.

Como resultado el A465 es capaz de mover el 100% de sus articulaciones, esta movilidad se vio limitada en comparación a la movilidad expuesta en el manual, con respecto a las articulaciones 1, 4, y 6 se redujo entre un 5 y 6%, para la articulación 5 en casi 10%, la mayor afectada fue la articulación 3 en un 50% y la menor la articulación 2 que se mantuvo el mismo rango, adicional a esto posee un error variable en cada una de ellas que ronda entre el 1 y 3% de su rango de movimiento total, en última instancia el controlador carece de algunas funcionalidades comunes como el control de la velocidad o una memoria independiente y añade otras como un simulador o el registro cinemático.

Adicionalmente los algoritmos y diagramas están pensados para que se pueda ir mejorando el controlador, en la tesis se abordó todo el procedimiento, esto abarco desde los fundamentos principales de la robótica y la cinemática, el análisis de los distintos dispositivos de microcontroladores del mercado y el criterio para elegir el más adecuado, teniendo como enfoque los actuadores y sensores con los que trabaja el robot.

También se expuso la metodología general para el desarrollo del controlador y, en base a ella, se realizó la circuitería y se programaron las interfaces para crear el prototipo lo que permitirá un desarrollo guiado al momento de enfrentar problemáticas similares y servirá para facilitar modificación del prototipo para corregir fallos, implementar mejoras u orientarlo a aplicaciones concretas.

5.2 Futuros proyectos

Se contempla que futuros trabajos en este ámbito podrían ser los siguientes:

- Corrección de los fallos y errores actuales del prototipo o, en su defecto, desarrollo de un nuevo prototipo contemplando lo aprendido en este trabajo, en este caso se recomendaría la implementación de un medio para la medición y control del amperaje empleándolo en cada articulación a la hora de moverse, esto debido a que se sospecha que a causa de la inclinación suceden la mayoría de las fallas, culpando así a un control de velocidad inadecuado.
- Adición de nuevos comandos como un bucle para repetir una secuencia, control de la velocidad de las articulaciones, un comando para crear un tiempo de espera, etcétera; gran parte de estos pueden ser programados únicamente en la interfaz de usuarios dejando de manera independiente el microcontrolador.
- Integración de otros sensores o actuadores empleando los recursos sobrantes en la placa del microcontrolador, en este caso se deberán de integrar nuevos caracteres especiales que sean recibidos y leídos desde el puerto serial, entre los principales se recomendaría la adición de un giroscopio para un mayor control del brazo o placas de presión que permitiesen el movimiento manual de la posición del robot.
- Reconexión de los componentes individuales en un sistema modular más estructurado para el fácil remplazo de estos, en un determinado caso sería extraer los componentes empleados del C500C y la circuitería desarrollada y colocarla en una nueva carcasa, donde también se agreguen fusibles, paro de emergencia, interruptor de encendido desde el Arduino, etcétera.

ANEXOS

Anexo A: Tabla de conexiones del CPC-24 para los motores

Pin #	Signal Name	Signal Description
1	Motor1+	Motor power ± 70 V @ 12 A max
2	Motor1-	Motor power return
3	Motor2+	Motor power ± 70 V @ 12 A max
4	Motor2-	Motor power return
5	Motor3+	Motor power ± 70 V @ 12 A max
6	Motor3-	Motor power return
7	N/C	
8	Motor4+	Motor power ± 30 V @ 3 A max
9	Motor4-	Motor power return
10	Motor5+	Motor power ± 30 V @ 3 A max
11	Motor5-	Motor power return
12	Motor6+	Motor power ± 30 V @ 3 A max
13	Motor6-	Motor power return
14	N/C	
15	N/C	
16	N/C	
17	N/C	
18	HomeSw1	Home switch, switch or Prox input, 12 V
19	HomeSw2	Home switch, switch or Prox input, 12 V
20	HomeSw3	Home switch, switch or Prox input, 12 V
21	N/C	
22	HomeSw4	Home switch, switch or Prox input, 12 V
23	HomeSw5	Home switch, switch or Prox input, 12 V
24	HomeSw6	Home switch, switch or Prox input, 12 V

Anexo B: Tabla de conexiones del CPC-57 para los encoders

Pin #	Signal Name	Signal Description
1	1A	RS422+, 200 Khz max pulse rate
2	1B	RS422+, 200 Khz max pulse rate
3	2A	RS422+, 200 Khz max pulse rate
4	2B	RS422+, 200 Khz max pulse rate
5	1A*	RS422-, 200 Khz max pulse rate
6	1B*	RS422-, 200 Khz max pulse rate
7	1Z	RS422+, 200 Khz max pulse rate
8	2Z	RS422+, 200 Khz max pulse rate
9	2A*	RS422-, 200 Khz max pulse rate
10	2B*	RS422-, 200 Khz max pulse rate
11,18	BRAKE	35 V @ 100 mA
12,13	SGM	± 15 V @ 300 mA
14	1Z*	RS422-, 200 Khz max pulse rate
15,16	Vcc	Encoder Supply +5 VDC @ 80 mA
17	Vcc	Encoder Supply +5 VDC @ 80 mA
19	6A	RS422+, 200 Khz max pulse rate
20	3A	RS422+, 200 Khz max pulse rate
21	3B	RS422+, 200 Khz max pulse rate
22	4A	RS422+, 200 Khz max pulse rate
23	4B	RS422+, 200 Khz max pulse rate
24,25	GND	Encoder digital return
26	6B	RS422+, 200 Khz max pulse rate
27	6Z	RS422+, 200 Khz max pulse rate
28	3Z	RS422+, 200 Khz max pulse rate
29	3A*	RS422-, 200 Khz max pulse rate
30	3B*	RS422-, 200 Khz max pulse rate

Pin #	Signal Name	Signal Description
31	4Z	RS422+, 200 Khz max pulse rate
32,33	BRAKERET*	return for brake supply
34	6A*	RS422-, 200 Khz max pulse rate
35	6B*	RS422-, 200 Khz max pulse rate
36	5A*	RS422-, 200 Khz max pulse rate
37	5B*	RS422-, 200 Khz max pulse rate
38	3Z*	RS422-, 200 Khz max pulse rate
39	4A*	RS422-, 200 Khz max pulse rate
40	GND	Encoder Digital return
41	2Z*	RS422-, 200 Khz max pulse rate
42	6Z*	RS422-, 200 Khz max pulse rate
43	5A	RS422+, 200 Khz max pulse rate
44	5Z*	RS422-, 200 Khz max pulse rate
45	4Z*	RS422-, 200 Khz max pulse rate
46	4B*	RS422-, 200 Khz max pulse rate
47	SGRIPTRQ	unused
48	SGRIPPOS	0-4.7 V
49	5B	RS422+, 200 Khz max pulse rate
50	5Z	RS422+, 200 Khz max pulse rate
51,52	Vcc	Encoder Supply +5 VDC @ 80 mA
53	S/A G+	Air solenoid 12 V @ 200 mA
54	S/A G-	Air solenoid return. This is switched by software control
55,57	GND	Encoder Digital return
56	N/C	

Anexo C: Códigos empleados en la programación del controlador

Durante el desarrollo del proyecto se crearon dos programas uno para la Interfaz Gráfica y otro para el Controlador, estos pueden ser encontrados en su totalidad y de manera gratuita en Gráfica alchacon.com, para el presente trabajo se añaden fragmentos que permiten una mayor comprensión de la lógica empleada en la programación.

C.1: Interrupciones del Arduino

```
void ai0() {
    if(micros()-lastMillis0>bounceDuration0){ //Filtro pasa-bajas
        //Determinar la direccion
        if(digitalRead(ENC_B[0])==LOW) { //La señal del otro encoder esta baja
            counter1++;
        }else{
            counter1--;
        }
        lastMillis0=micros();//reestablecer valor para filtro
    }
}

void ail() {
    if(micros()-lastMillis1>bounceDuration1){ //Filtro pasa-bajas
        //Determinar la direccion
        if(digitalRead(ENC_A[0])==LOW) { //La señal del otro encoder esta baja
            counter1--;
        }else{
            counter1++;
        }
        lastMillis1=micros();//reestablecer valor para filtro
    }
}
```

C.2: Actualizar la posición

```
void PosicionEncoder1() {  
    //Valor de la posición actual  
    if( counter1 != temp1 ){  
        temp1 = counter1;  
        posicionActual[0] = counter1;  
    }  
}
```

C.3: Señales del controlador al puente H

```
void ReducirAngulo(int velocidad, int motor){  
    //Velocidad  
    analogWrite(Pulsos[motor], velocidad);  
    //Senal  
    digitalWrite(DIR[motor], HIGH);  
}  
  
void AumentarAngulo(int velocidad, int motor){  
    //Velocidad  
    analogWrite(Pulsos[motor], velocidad);  
    //Senal  
    digitalWrite(DIR[motor], LOW);  
}  
  
void Detener(int motor){  
    //Velocidad  
    analogWrite(Pulsos[motor], 0);  
}
```

C.4: Función para mandar a posición deseada

```
void MandarPosicionDeseada(int motor, int voltage, int velocidad, int espera){
    ///Evitar interferencia de movimientos entre articulaciones
    counter1 = posicionActual[0];
    ...//
    counter6 = posicionActual[5];

    //Iniciar tiempo 2
    tiempo2 = millis();
    PosAnterior = 0;

    boolean MotorEnPosicion = false;
    while(!MotorEnPosicion){
        //Actualizar posición
        switch(motor){
            case 0:
                PosicionEncoder1();
                counter2 = posicionActual[1];
                counter3 = posicionActual[2];
                counter4 = posicionActual[3];
                counter5 = posicionActual[4];
                counter6 = posicionActual[5];
                break;
            ...
            case 5:
                PosicionEncoder6();
                counter1 = posicionActual[0];
                counter2 = posicionActual[1];
                counter3 = posicionActual[2];
                counter4 = posicionActual[3];
                counter5 = posicionActual[4];
                break;
        }

        //Motor en posición
        if((posicionActual[motor] <= (posicionDeseada[motor]+rango)) &&
(posicionActual[motor] >= (posicionDeseada[motor]-rango))){
            Detener(motor);
            posicionActual[motor] = posicionDeseada[motor];

            ///Evitar interferencia de movimientos entre articulaciones
            counter1 = posicionActual[0];
            ...
            counter6 = posicionActual[5];
        }
    }
}
```



```

    MotorEnPosicion = true;
}
//Motor atras de la posición deseada
else if(posicionDeseada[motor] > posicionActual[motor]){
    ReducirAngulo(voltage,motor);
}
//Motor delante de la posición deseada
else if(posicionDeseada[motor] < posicionActual[motor]){
    AumentarAngulo(voltage,motor);
}

//Leer posición actual y tiempo
PosActual = posicionActual[motor];
tiempo1 = millis();

//Calcular tiempo transcurrido
unsigned long tiempo = tiempo1 - tiempo2;

//Calcular distancia recorrida
int distancia = PosActual- PosAnterior;
int velocidadActual = abs(distancia)/tiempo;

if(tiempo >= espera){
    int distanciaFaltante = PosActual -posicionDeseada[motor];
    if(abs(distanciaFaltante) > 15*555){
        //Regular la velocidad cambiando el voltage
        if((velocidad > velocidadActual)&& (voltage < 200)){
            voltage += 5;
        }
        else if((velocidad < velocidadActual) && (velocidadActual > 2)){
            voltage -= 5;
        }
    }
    else if((abs(distanciaFaltante) < 15*555) && (abs(distanciaFaltante) >
5*555) && (velocidadActual > 3) && (velocidadActual < 2)){
        //Serial.println("Disminuyendo velocidad");
        voltage -=5;
    }

    PosAnterior = PosActual;
    tiempo2 = tiempo1;
}
}
}

```

C.5: Confirmación de la comunicación Arduino

```
void loop() {
  //Revisar puerto serial
  if (Serial.available() > 0) {
    data = Serial.read();

    switch (data) {
      case 'a': //Comprobacion de conexión
        Serial.println("1");
        digitalWrite(LED_PIN, HIGH);
        break;
    }
  }
  ...
}
```

C.6: Confirmación de la comunicación Java

```
public static void conectarse() throws InterruptedException, RuntimeException{
  ...

  //Tratar de mandar y recibir un dato
  String txt1 = "JOptionPane showMessageDialog example";
  try{
    //Mandar dato de conexión
    arduino.serialWrite('a');
    //Recibir conexión
    dato = arduino.serialRead(1);
    //Pasar dato a double para compararlo
    comprobacion = Double.parseDouble(dato);
  }
  catch (RuntimeException ex){ //Avisar que no se conecto
    String txt2 = "Coneccion No Establecida";
    JOptionPane.showMessageDialog(new JFrame(txt1), txt2);
  }

  //Comprobar que arduino mando el valor correcto
  if (comprobacion == 1) {
    String txt2 = "Coneccion Establecida";
    JOptionPane.showMessageDialog(new JFrame(txt1), txt2);
    conectado = true; //detener bucle avisando que ya se conecto
    ventanaInicial.setVisible(false); //Ocultar ventana de conectarse
  }
  else{
    String txt2 = "Ingrese un nuevo puerto o baudios";
    JOptionPane.showMessageDialog(new JFrame(txt1), txt2);
  }
}
```

```

        //Volver a pasar los datos a false para esperar datos nuevos
        ventanaInicial.datos = false;
    }
}

```

C.7: Envío de caracteres a Arduino desde Java

```

arduino.serialWrite('k');//Mandar a posición

    dato = "0";
    while(Double.parseDouble(dato) != 23){
        //Recibir conexión
        dato = arduino.serialRead(1);

        if(!(dato != null && dato.length() > 0)){
            dato = "0";
        }
    }
    //Pasar dato a double para compararlo
    System.out.println(Double.parseDouble(dato));
    //Terminar con el Home
    if(Double.parseDouble(dato) == 23){

    }
}

```

C.8: Envío de una nueva posición desde Java

```

public static void mandarPos(int union, int ángulo){
    if(conectado){
        //Obtener digitos del ángulo
        int centena = ángulo/100;
        int decena = (ángulo-centena*100)/10;
        int unidad = (ángulo - centena*100 - decena*10)/1;

        char c = (char)(centena+'0');
        char d = (char)(decena+'0');
        char u = (char)(unidad+'0');

        //Pasar la union a char
        char joint = (char)((union-1)+'0');

        //Mandar a arduino
        arduino.serialWrite('f');//Indicar que se cambiara una posición
        arduino.serialWrite('j');//Indicar que se mandara la union
        arduino.serialWrite(joint);//Indicar de que union se trata
    }
}

```

```

        arduino.serialWrite('c');//Indicar que se mandaran las centenas
        arduino.serialWrite(c);//Indicar las centenas
        arduino.serialWrite('d');//Indicar que se mandaran las decenas
        arduino.serialWrite(d);//Indicar las decenas
        arduino.serialWrite('u');//Indicar que se mandaran las unidades
        arduino.serialWrite(u);//Indicar las unidades
        arduino.serialWrite('x');//Indicar que ya se mando todo
    }
}

```

C.9: Lectura de una nueva posición desde Arduino

```

void LeerPosicionDeseada() {
    //Serial.println("Entro a la funcion");

    int cambio[] = {0,0,0,0}; //Union, Centena, Decena, Unidad
    int indice = 0; //Seleccionar union o ángulo

    //Recibir valores de java
    boolean PosicionLeida = false;
    while (!PosicionLeida) {
        if (Serial.available() > 0) {
            data = Serial.read();

            switch (data) {

                case 'j':
                    //Serial.println("Union");
                    indice = 0;
                    break;

                case 'c':
                    //Serial.println("Centena");
                    indice = 1;
                    break;

                case 'd':
                    //Serial.println("Decena");
                    indice = 2;
                    break;

                case 'u':
                    //Serial.println("Unidad");
                    indice = 3;
                    break;
            }
        }
    }
}

```

```
case '0':
    //Serial.println("0");
    cambio[indice] = 0;
    break;

case '1':
    //Serial.println("1");
    cambio[indice] = 1;
    break;

case '2':
    //Serial.println("2");
    cambio[indice] = 2;
    break;

case '3':
    //Serial.println("3");
    cambio[indice] = 3;
    break;

case '4':
    //Serial.println("4");
    cambio[indice] = 4;
    break;

case '5':
    //Serial.println("5");
    cambio[indice] = 5;
    break;

case '6':
    //Serial.println("6");
    cambio[indice] = 6;
    break;

case '7':
    //Serial.println("7");
    cambio[indice] = 7;
    break;

case '8':
    //Serial.println("8");
    cambio[indice] = 8;
    break;

case '9':
    //Serial.println("9");
```

```

        cambio[indice] = 9;
        break;

    case 'x':
        //Serial.println("Salir");
        PosicionLeida = true;
        break;

    default:
        // if nothing else matches, do the default
        // default is optional
        break;
    }
}

//Calcular posición deseada
int posicionRecibida = cambio[1]*100 + cambio[2]*10 + cambio[3];

//Actualizar valor deseado
posicionDeseada[cambio[0]] = -1*posicionRecibida*555;
}
}

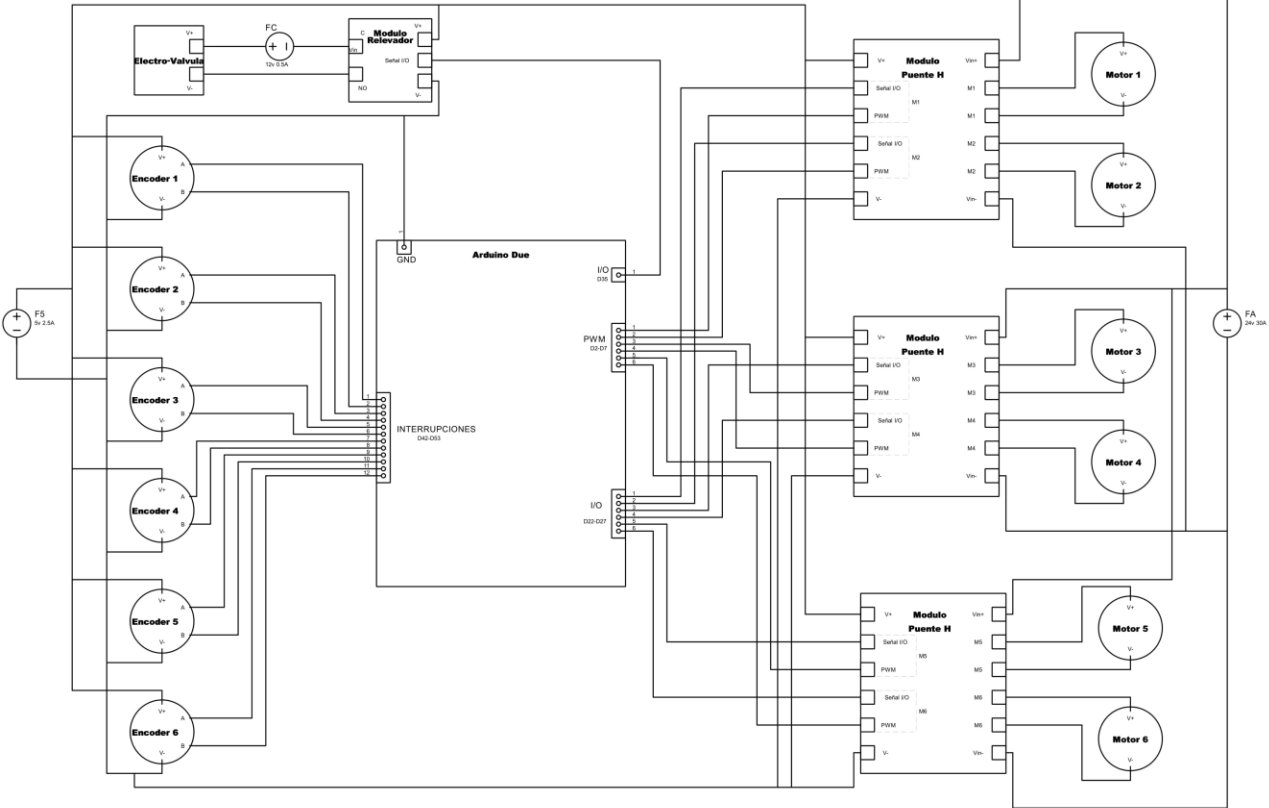
```

Anexo D: Código de letras para la comunicación Java-Arduino

Letra	Función Arduino	Función Java
a	Encender LED y enviar de Recibido	Comprobar conexión con Arduino
b	Mover motores hasta tocar alcanzar Home y Guardar ángulos como posición 0	Indicar que los motores se muevan a Home
c	Indicar que el siguiente valor corresponde a las centenas para el ángulo y guardarlo	Indicar que se enviará la centena del ángulo
d	Indicar que el siguiente valor corresponde a las decenas para el ángulo y guardarlo	Indicar que se enviará la decena del ángulo
e	Entrar en modo de “Mando de posición” y esperar articulación	Indicar que se solicitara una posición
f	Entrar en modo de “Recibir posición” y esperar articulación	Indicar que se desea una nueva posición
g	Mandar motores a sus posiciones deseadas	Indicar que se manden los motores a sus posiciones deseadas
h	Establece la posición actual como Home 175,90,200,180,105,180	Indicar que se emplea la posición actual como Home
i	Establece la posición actual como Home 175,90,110,180,105,180	Indicar que se establezcan valores para los ángulos para buscar el Home
j	Indicar que el siguiente valor corresponde a la unión	Indicar que se enviará la unión
k	Establece las posiciones deseadas como las actuales	Indica que se establezcan las posiciones deseadas como actuales, es usado para indicar la última posición guardada al iniciar el programa

p	Invierte la señal de salida para cerrar o abrir la pinza	Envía la señal para abrir o cerrar la pinza
r	Envía las posiciones actuales almacenadas en el Arduino	Solicita las posiciones actuales del Arduino
u	Indicar que el siguiente valor corresponde a las unidades para el ángulo y guardarlo	Indicar que se enviará la unidad del ángulo
x	Dar por terminado la recepción de la nueva posición	Indicar que ya se ha enviado la información de la nueva posición

Anexo E: Representación esquemática completa del circuito eléctrico



REFERENCIAS BIBLIOGRAFICAS

- [1] Designs, W., 2020. CRS Robotics A465 Robot Arm With C500C Controller. [online] Americanlaboratorytrading.com. Available at: <https://americanlaboratorytrading.com/lab-equipment-products/crs-robotics-a465-robot-arm-with-c500c-controller_11572> [Accessed 13 November 2020].
- [2] B. Muhammad, O. Muhammad, M. Awais, A. Norman, "Design and Control of 6 DOF Robotic Manipulator", University of Engineering and Technology Lahore Faisalabad Campus, 2018.
- [3] F. J. García Gutiérrez, "Desarrollo y construcción de una tarjeta para la adquisición de datos y control de robots cooperativos", Universidad Nacional Autónoma de México, Coyoacán, 2016.
- [4] C. Campos Caldera, J. E. Alderete Alderete, J. C. Corrujedo Lazcano, J. C. Soto Armenta, "Reingeniería del Controlador de un Brazo Robótico de 6 g.d.l", *Revista de Ingeniería Biomédica y Biotecnología*, Vol. 2, no. 4, Junio, pp. 22-33, 2018.
- [5] C. F. Salinas García, "Modelo Cinemático Directo Del Robot CRS Robotics A465", Instituto Tecnológico De Estudios Superiores De La Región Carbonífera, Villa de Agujita, 2011.
- [6] C.J.A.Tiburcio Mejia, "Reparación de brazo robótico CRS A465 e interfaz gráfica en LabView PIC18F4550", Universidad Politécnica de Sinaloa, 2016.
- [7] J. E. Palomares Ruiz, "Modelación Cinemática Del Robot CRS A465, Utilizando El Álgebra De Cuaterniones", Universidad Nacional Autónoma de México, 2006.
- [8] W. Bolton, *Mechatronics Electronic Control System In Mechanical And Electrical Engineering*, Pearson, 2015

- [9] I. L. Kosow, Máquinas eléctricas y transformadores, Irving L. Prentice-Hall Hispanoamericana, 1993.
- [10] M. W. Spong, S. Hutchinson, M. Vidyasagar, Robot Modeling and Control, Jhon Wiley & Sons, 2005
- [11] A. Lazinica, Eds., Mobile Robots Towards New Applications, pro literatur Verlag, 2006.
- [12] M. A. Mazidi, J. G. Mazidi, R. D. McKinlay, “The 8051 Microcontroller and Embedded System”, Pearson, 2006.
- [13] Arduino.cc. 2020. Arduino - Home. [online] Available at: <<https://www.arduino.cc/>> [Accessed 13 November 2020].
- [14] A. Benito Herranz, “Desarrollo de aplicaciones para IoT con el módulo ESP32”, Universidad de Alcalá Escuela Politécnica Superior, 2019
- [15] Ti.com. 2020. Hercules Arm Cortex-R Functional Safety Mcus | Overview | TI.Com. [online] Available at: <<https://www.ti.com/microcontrollers/hercules-safety-mcus/overview.html>> [Accessed 13 November 2020].
- [16] Arduino.cc. 2020. Arduino - Arduinoboarddue. [online] Available at: <<https://www.arduino.cc/en/pmwiki.php?n=Main/ArduinoBoardDue>> [Accessed 13 November 2020].
- [17] A. Barrientos, L. F. Peñin, C. Balaguer, R. Aracil, Fundamentos de Robótica, Mc Graw Hill, 2007.
- [18] CRS Robotics Corporation, A465 Robot Arm User Guide For use with a C500C Controller, CRS, 2000

[19] Sistema Mecatronico Para Interceptar Y Capturar Objetos En Movimiento Vía Robot Industrial, Unidad Profesional Interdisciplinaria en Ingeniería y Tecnologías Avanzadas.

[20] CRS Robotics Corporation, C500C Controller User Guide, CRS, 2000